

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

```
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      FFFFFFFFFF      IIIIIII
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      FFFFFFFFFF      IIIIIII
BB          BB  AA          AA  SS          SS          RRRRRRRR      SS          SS          TT          SS          FFFFFFFFFF      IIIIIII
BB          BB  AA          AA  SS          SS          RRRRRRRR      SS          SS          TT          SS          FFFFFFFFFF      IIIIIII
BB          BB  AA          AA  SS          SS          RRRRRRRR      SS          SS          TT          SS          FFFFFFFFFF      IIIIIII
BB          BB  AA          AA  SS          SS          RRRRRRRR      SS          SS          TT          SS          FFFFFFFFFF      IIIIIII
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      RRRRRRRR      SSSSSSS      SSSSSSS      TT          TT          FFFFFFFFFF      IIIIIII
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      RRRRRRRR      SSSSSSS      SSSSSSS      TT          TT          FFFFFFFFFF      IIIIIII
BB          BB  AAAAAAAAAA      SS          SS          RR      RR      SS          SS          FF          FF          IIIIIII
BB          BB  AAAAAAAAAA      SS          SS          RR      RR      SS          SS          FF          FF          IIIIIII
BB          BB  AA          AA  SSSSSSS      SSSSSSS      RR      RR      SS          SS          FF          FF          IIIIIII
BB          BB  AA          AA  SSSSSSS      SSSSSSS      RR      RR      SS          SS          FF          FF          IIIIIII
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      RR      RR      SSSSSSSS      SSSSSSSS      FF          FF          IIIIIII
BBBBBBBBB      AA          AA  SSSSSSS      SSSSSSS      RR      RR      SSSSSSSS      SSSSSSSS      FF          FF          IIIIIII

LL          IIIIIII      SSSSSSSS
LL          IIIIIII      SSSSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SSSSSS
LL          II          SSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LLLLLLLLLLL      IIIIIII      SSSSSSSS
LLLLLLLLLLL      IIIIIII      SSSSSSSS
```



```
1 0001 0 MODULE BASRSTS_FIELD (          ! FIELD statement
2 0002 0      IDENT = '1-023'          ! File: BASRSTSFI.B32 Edit: MDL1023
3 0003 0      ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 *   ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 *   TRANSFERRED.
18 0018 1 *
19 0019 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 *   CORPORATION.
22 0022 1 *
23 0023 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1
30 0030 1 **
31 0031 1 FACILITY: VAX-11 BASIC Miscellaneous
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1     This module contains the RSTS-compatible FIELD functions.
36 0036 1     A FIELD variable is semi-interpreted, and routines in this
37 0037 1     module "declare" such a variable, copy data to and from it,
38 0038 1     and purge it when the block it was in is exited.
39 0039 1
40 0040 1 ENVIRONMENT: VAX-11 User Mode
41 0041 1
42 0042 1 AUTHOR: John Sauter, CREATION DATE: 27-FEB-1979
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 1-001 - Original. JBS 27-FEB-1979
47 0047 1 1-002 - Rearrange FIELD SET so that the compiler can call it conveniently
48 0048 1         once for a FIELD statement. JBS 01-MAR-1979
49 0049 1 1-003 - Add a statement type parameter to FIELD COPY. JBS 02-APR-1979
50 0050 1 1-004 - Correct STR$COPY to STR$COPY_DX. JBS 03-APR-1979
51 0051 1 1-005 - Re-order some parameters to make things easier on the BASIC-PLUS-2
52 0052 1         compiler. JBS 18-MAY-1979
53 0053 1 1-006 - Today the compiler began producing code for the FIELD
54 0054 1         statement, so begin debugging. JBS 22-MAY-1979
55 0055 1 1-007 - Add OPEN, CLOSE and KILL entry points. JBS 24-MAY-1979
56 0056 1 1-008 - Complete coding of the new entry points. JBS 25-MAY-1979
57 0057 1 1-009 - Add BAS$$FIELD_INIT. JBS 04-JUN-1979
```

```

: 58      0058 1 : 1-010 - Fix a bug in KILL which made it run forever. JBS 07-JUN-1979
: 59      0059 1 : 1-011 - Allow a virtual array to be fielded, but only if it is used
: 60      0060 1 : exclusively for block I/O. JBS 22-JUN-1979
: 61      0061 1 : 1-012 - Add BAS$FIELD COP R. JBS 13-JUL-1979
: 62      0062 1 : 1-013 - Change calls to STR$COPY. JBS 16-JUL-1979
: 63      0063 1 : 1-014 - Set up ISB$A USER_FP. JBS 25-JUL-1979
: 64      0064 1 : 1-015 - Call STR$FREE1 DX only for variables not already FIELDed.
: 65      0065 1 : JBS 02-AUG-1979
: 66      0066 1 : 1-016 - Call BAS$$CB GET, so we don't have to be in the sharable
: 67      0067 1 : library. JBS 22-AUG-1979
: 68      0068 1 : 1-017 - Make negative string lengths back up the address but set
: 69      0069 1 : the variable's length to zero. JBS 28-FEB-1980
: 70      0070 1 : 1-018 - When opening a file, validate variables which may have been
: 71      0071 1 : invalidated by the implied close. JBS 23-MAY-1980
: 72      0072 1 : 1-019 - Add 2 to lub$klun_min before calling bas$$cb_push to force an
: 73      0073 1 : error when -7 or -8 LUNs are used, at the same time put the code that
: 74      0074 1 : opens channel 0 for upgrading bas$field_set to the level of the other
: 75      0075 1 : I/O support routines. FM 17-sep-80
: 76      0076 1 : 1-020 - Make SYM$Q_ROOT global so that BAS$CLOSE can access it.
: 77      0077 1 : PL 2-JUN-81
: 78      0078 1 : 1-021 - BAS$CLOSE will not need SYM$Q_ROOT after all, so make it OWN again.
: 79      0079 1 : PL 16-Jun-81
: 80      0080 1 : 1-022 - Undo 19. We can now do I/O to #0, because BAS$PUT will use foreign
: 81      0081 1 : buffer mechanism to do PUTs to #0. FM 9-JUL-81.
: 82      0082 1 : 1-023 - set LUB$V_FIELD_USE when a FIELD statement is executed, so that
: 83      0083 1 : BAS$CLOSE can tell if there is FIELDing on the channel. Clear it
: 84      0084 1 : when the channel is closed. MDL 29-Mar-1984
: 85      0085 1 : --
: 86      0086 1 :
: 87      0087 1 : <BLF/PAGE>

```



```

89 0088 1 |
90 0089 1 | SWITCHES:
91 0090 1 |
92 0091 1 |
93 0092 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
94 0093 1 |
95 0094 1 |
96 0095 1 | LINKAGES:
97 0096 1 |
98 0097 1 |
99 0098 1 | REQUIRE 'RTLIN:OTSLNK';           ! Define Linkages
100 0527 1 |
101 0528 1 |
102 0529 1 | TABLE OF CONTENTS:
103 0530 1 |
104 0531 1 |
105 0532 1 | FORWARD ROUTINE
106 0533 1 |     BASSFIELD_SET : NOVALUE,       ! Process a FIELD statement
107 0534 1 |     BASSFIELD_VAR : CALL_CCB NOVALUE, ! Declare a FIELD variable
108 0535 1 |     BASSFIELD_CLEAR : NOVALUE,     ! Undeclare a FIELD variable
109 0536 1 |     BASSFIELD_COPY : NOVALUE,      ! Reference such a variable
110 0537 1 |     BASSFIELD_COPY_R : NOVALUE,    ! Reference such a variable
111 0538 1 |     BASSFIELD_PURGE : NOVALUE,     ! Purge field variables
112 0539 1 |     BASSFIELD_OPEN : NOVALUE,      ! A file was just opened
113 0540 1 |     BASSFIELD_CLOSE : NOVALUE,     ! A file was just closed
114 0541 1 |     BASSFIELD_KILL : CALL_CCB NOVALUE, ! CLOSE appendage
115 0542 1 |     BASSFIELD_INIT : NOVALUE;      ! Initialize the FIELD list
116 0543 1 |
117 0544 1 |
118 0545 1 | INCLUDE FILES:
119 0546 1 |
120 0547 1 |
121 0548 1 | REQUIRE 'RTLIN:RTLPSECT';         ! Macros for defining psects
122 0643 1 |
123 0644 1 | REQUIRE 'RTLML:OTSLUB';          ! Logical unit block definitions
124 0784 1 |
125 0785 1 | REQUIRE 'RTLML:OTSISB';          ! ISB definitions
126 0953 1 |
127 0954 1 | REQUIRE 'RTLIN:BASFRAME';        ! BASIC frame structure
128 1157 1 |
129 1158 1 | LIBRARY 'RTLSTARLE';             ! System definitions
130 1159 1 |
131 1160 1 |
132 1161 1 | MACROS:
133 1162 1 |
134 1163 1 |     NONE
135 1164 1 |
136 1165 1 | EQUATED SYMBOLS:
137 1166 1 |
138 1167 1 |
139 1168 1 | LITERAL
140 1169 1 |     STMT_TYPE_LSET = 0,           ! LSET statement
141 1170 1 |     STMT_TYPE_RSET = 1;          ! RSET statement
142 1171 1 |
143 1172 1 |
144 1173 1 | PSECTS:
145 1174 1 |
```

```
146 1175 1 DECLARE_PSECTS (BAS);           ! Declare psepts for BAS$ facility
147 1176 1
148 1177 1 ! OWN STORAGE:
149 1178 1
150 1179 1
151 1180 1 OWN
152 1181 1     SYMSQ_ROOT : VECTOR [2] INITIAL (0, 0);   ! Root for symbol table
153 1182 1
154 1183 1
155 1184 1 ! EXTERNAL REFERENCES:
156 1185 1
157 1186 1
158 1187 1 EXTERNAL ROUTINE
159 1188 1     BAS$$STOP : NOVALUE,           ! Signal fatal error
160 1189 1     BAS$$STOP IO : NOVALUE,      ! Signal fatal I/O error
161 1190 1     LIB$GET_VM,                 ! Get virtual memory
162 1191 1     LIB$FREE_VM,                 ! Free virtual memory
163 1192 1     BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE, ! Load register CCB
164 1193 1     BAS$$CB_POP : JSB_CB_POP NOVALUE,  ! Done with register CCB
165 1194 1     BAS$$CB_GET : JSB_CB_GET NOVALUE,  ! Load CCB with current LUB
166 1195 1     STR$COPY_DX,                ! Copy a string (LSET)
167 1196 1     STR$FREE_DX,                ! Free a string
168 1197 1     BAS$RSET,                  ! Copy a string (RSET)
169 1198 1     BAS$$OPEN_ZERO : NOVALUE;       ! Open channel 0
170 1199 1
171 1200 1 !+
172 1201 1 ! The following are the error codes used in this module.
173 1202 1 !-
174 1203 1
175 1204 1 EXTERNAL LITERAL
176 1205 1     BAS$K_MAXMEMEXC : UNSIGNED (8),      ! Maximum memory exceeded
177 1206 1     BAS$K_PROLOSSOR : UNSIGNED (8),      ! Program lost, sorry
178 1207 1     BAS$K_IO_CHANOT : UNSIGNED (8),      ! I/O channel not open
179 1208 1     BAS$K_ILILLACC : UNSIGNED (8),      ! Illegal or illogical access
180 1209 1     BAS$K_FIEOVEBUF : UNSIGNED (8),      ! Field overflows buffer
181 1210 1     BAS$K_ILLFIEVAR : UNSIGNED (8),      ! Illegal FIELD variable
182 1211 1     BAS$K_ILLIO_CHA : UNSIGNED (8);      ! Illegal I/O channel
183 1212 1
184 1213 1 !<BLF/PAGE>
```

```

: 186      1214 1  !+
: 187      1215 1  ! The following field represents a symbol table entry.
: 188      1216 1  !-
: 189      1217 1
: 190      1218 1 FIELD
: 191      1219 1     BASFIELD_SYM =
: 192      1220 1     SET
: 193      1221 1     SYMSA_NEXT = [0, 0, %BPADDR, 0],      ! Next symbol table entry
: 194      1222 1     SYMSA_PREV = [%UPVAL, 0, %BPADDR, 0], ! Previous entry
: 195      1223 1     SYMSL_CHAN = [%UPVAL*2, 0, %BPVAL, 0], ! I/O channel
: 196      1224 1     SYMSL_OFFSET = [%UPVAL*3, 0, %BPVAL, 0], ! Offset into I/O buffer
: 197      1225 1     SYMSL_LEN = [%UPVAL*4, 0, %BPVAL, 0],  ! Number of bytes referenced
: 198      1226 1     SYMSL_DECL = [%UPVAL*5, 0, %BPVAL, 0], ! Scope of the declaration
: 199      1227 1     SYMSA_VAR = [%UPVAL*6, 0, %BPADDR, 0], ! Address of descriptor
: 200      1228 1     SYMSV_INVALID = [%UPVAL*7, 0, 1, 0],  ! "Invalid" bit
: 201      1229 1     TES;
: 202      1230 1
: 203      1231 1 LITERAL
: 204      1232 1     SYMSK_LENGTH = %UPVAL*8;                ! Number of bytes to allocate
: 205      1233 1

```



```
207 1234 1 GLOBAL ROUTINE BASSFIELD_SET          ! Execute a FIELD statement
208 1235 1 : NOVALUE =
209 1236 1
210 1237 1 ++
211 1238 1 FUNCTIONAL DESCRIPTION:
212 1239 1
213 1240 1     Execute a FIELD statement. The compiler pushes all of the
214 1241 1     variables in the FIELD statement from right to left and then
215 1242 1     calls this routine. As a result, the formal parameters are
216 1243 1     arranged rather strangely. This routine goes through them
217 1244 1     calling BASSFIELD_VAR for each variable.
218 1245 1
219 1246 1     The FIELD statement is formatted as follows:
220 1247 1
221 1248 1     FIELD #chan, exp BY var, exp BY var, ...
222 1249 1
223 1250 1 FORMAL PARAMETERS:
224 1251 1
225 1252 1     The formal parameters are rather strange, for the convenience of
226 1253 1     the compiler. Because the compiler likes to push parameters in the
227 1254 1     order in which it encounters them, the pairs of optional parameters
228 1255 1     are first, followed by the fixed parameters. The list below is of
229 1256 1     the parameters in reverse order.
230 1257 1
231 1258 1     CHAN.rl.v      An I/O channel. Must be open.
232 1259 1     DECL.rl.v      An indication of the scope of the declaration
233 1260 1                  of the variable. This is a pointer to the major
234 1261 1                  frame (R11) if the variable is in the scope of
235 1262 1                  the major procedure, or a pointer to the minor
236 1263 1                  frame (R10) if the variable is in the scope of
237 1264 1                  a DEF.
238 1265 1
239 1266 1     The following two parameters can be repeated as often as required.
240 1267 1
241 1268 1     LEN.rl.v       The number of bytes referenced by the variable
242 1269 1     VAR.wt.d       The variable. Since references to it ignore its
243 1270 1                  previous (non-FIELD) contents, we free it here.
244 1271 1
245 1272 1 IMPLICIT INPUTS:
246 1273 1
247 1274 1     SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.
248 1275 1     LUBSV_VA_USE   If this bit is set, the file has been used
249 1276 1                  with a virtual array, and so cannot be used
250 1277 1                  with the FIELD statement.
251 1278 1
252 1279 1 IMPLICIT OUTPUTS:
253 1280 1
254 1281 1     SYMSQ_ROOT.mq  This bit is set to prevent the file from
255 1282 1     LUBSV_BLK_USE  being used with a virtual array.
256 1283 1
257 1284 1     LUBSV_FIELD_USE for this channel, set to 1
258 1285 1
259 1286 1 ROUTINE VALUE:
260 1287 1 COMPLETION CODES:
261 1288 1
262 1289 1     NONE
263 1290 1
```



```
264 1291 1 SIDE EFFECTS:
265 1292 1
266 1293 1 Adds symbols to the interpreter's symbol table, and/or modifies
267 1294 1 symbols already there.
268 1295 1
269 1296 1 --
270 1297 1
271 1298 1 BEGIN
272 1299 1
273 1300 1 GLOBAL REGISTER
274 1301 1 CCB = K_CCB_REG : REF BLOCK [, BYTE];
275 1302 1
276 1303 1 BUILTIN
277 1304 1 FP,
278 1305 1 ACTUALCOUNT,
279 1306 1 ACTUALPARAMETER;
280 1307 1
281 1308 1 LOCAL
282 1309 1 FMP : REF BLOCK [, BYTE],
283 1310 1 NUMARGS,
284 1311 1 DECL,
285 1312 1 CHAN,
286 1313 1 LEN,
287 1314 1 VAR,
288 1315 1 OFFSET,
289 1316 1 LUN_NO,
290 1317 1 RBF,
291 1318 1 RSZ;
292 1319 1
293 1320 1 FMP = .FP;
294 1321 1 +
295 1322 1 Start at the beginning of the buffer.
296 1323 1 -
297 1324 1 OFFSET = 0;
298 1325 1 +
299 1326 1 Fetch, from the peculiar argument list, the DECL and CHAN parameters.
300 1327 1 -
301 1328 1 NUMARGS = ACTUALCOUNT ();
302 1329 1 CHAN = ACTUALPARAMETER (.NUMARGS);
303 1330 1 DECL = ACTUALPARAMETER (.NUMARGS - 1);
304 1331 1 +
305 1332 1 Compute the logical unit number corresponding to the channel, and
306 1333 1 validate it.
307 1334 1 -
308 1335 1
309 1336 1 IF (.CHAN LSS 0) THEN BAS$$STOP (BAS$K_ILLIO_CHA);
310 1337 1
311 1338 1 LUN_NO = (IF (.CHAN EQL 0) THEN LUB$K_LUN_INPU ELSE .CHAN);
312 1339 1 BAS$$CB PUSH (.LUN_NO, LUB$K_ILUN_MINT);
313 1340 1 CCB [ISB$A_USER_FP] = .FMP [SF$A_SAVE_FP];
314 1341 1
315 1342 1 IF ( NOT .CCB [LUB$V_OPENED])
316 1343 1 THEN
317 1344 1 IF (.LUN_NO LSS 0)
318 1345 1 THEN
319 1346 1 BEGIN
320 1347 1 BAS$$OPEN_ZERO (.FMP [SF$A_SAVE_FP] );
```

```
321      1348      3
322      1349      3
323      1350      3
324      1351      3
325      1352      3
326      1353      3
327      1354      3
328      1355      3
329      1356      3
330      1357      3
331      1358      3
332      1359      3
333      1360      3
334      1361      3
335      1362      3
336      1363      3
337      1364      3
338      1365      3
339      1366      3
340      1367      3
341      1368      3
342      1369      3
343      1370      3
344      1371      3
345      1372      3
346      1373      3
347      1374      3
348      1375      3
349      1376      3
350      1377      3
351      1378      3
352      1379      3
353      1380      3
354      1381      3
355      1382      3
356      1383      3
357      1384      3
358      1385      3
359      1386      3
360      1387      3
361      1388      3
362      1389      3
363      1390      3
364      1391      3
365      1392      3
366      1393      3
367      1394      3
368      1395      3
369      1396      3
370      1397      1

      END
    ELSE
      BEGIN
        BAS$$STOP_IO (BAS$K_IO_CHANOT);
      END;

      IF (.CCB [LUB$A_CLOSE] EQ 0) THEN CCB [LUB$A_CLOSE] = BAS$$FIELD_KILL;
      IF (.CCB [LUB$A_CLOSE] NEQ BAS$$FIELD_KILL) THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
      CCB [LUB$V_BLK_USE] = 1;
      CCB [LUB$V_FIELD_USE] = 1;
      IF (.CCB [LUB$V_VA_USE]) THEN BAS$$STOP_IO (BAS$K_ILLILLACC);

      + Fetch the buffer address and length for later use
      -
      RBF = .CCB [LUB$A_RBUF_ADR];
      RSZ = .CCB [LUB$W_RBUF_SIZE];

      + Go through the optional arguments, associating each variable with
      - its place in the I/O buffer. We scan the variables from left to
      right in the FIELD statement in case the same variable appears twice:
      it should end up with its right-most association.
      -

      DECR ARGNO FROM .NUMARGS - 2 TO 1 BY 2 DO
        BEGIN
          LOCAL
            LEN,
            VAR;

          LEN = ACTUALPARAMETER (.ARGNO);
          VAR = ACTUALPARAMETER (.ARGNO - 1);

          IF (.OFFSET + .LEN GTR .RSZ) THEN BAS$$STOP_IO (BAS$K_FIEOVEBUF);

          BAS$FIELD_VAR (.CHAN, .OFFSET, .LEN, .DECL, .VAR, .RBF);
          OFFSET = .OFFSET + .LEN;
        END;

      + We are done with register CCB
      -
      BAS$$CB_POP ();
      RETURN;
    END;
```

! end of BAS\$FIELD\_SET

```
.TITLE BASRSTS_FIELD
.IDENT \1-023\
.PSECT _BAS$DATA,NOEXE, PIC,2
```



00000000 00000000 00000 SYMSQ\_ROOT:

.LONG 0, 0

.EXTRN BAS\$\$STOP, BAS\$\$STOP\_IO  
.EXTRN LIB\$GET\_VM, LIB\$FREE\_VM  
.EXTRN BAS\$\$CB\_PUSH, BAS\$\$CB\_POP  
.EXTRN BAS\$\$CB\_GET, STR\$COPY\_DX  
.EXTRN STR\$FREE1\_DX, BASRSET  
.EXTRN BAS\$\$OPEN\_ZERO, BASK\_MAXMEMEXC  
.EXTRN BASK\_PROCESSOR  
.EXTRN BASK\_IO\_CHANOT  
.EXTRN BASK\_ILLLACC  
.EXTRN BASK\_FIEOVEBUF  
.EXTRN BASK\_ILLFIEVAR  
.EXTRN BASK\_ILLIO\_CHA

.PSECT \_BAS\$CODE, NOWRT, SHR, PIC, 2

			OFFC	00000			
					.ENTRY	BAS\$FIELD_SET, Save R2,R3,R4,R5,R6,R7,R8,-	1234
						R9,R10,R11	
					MOVAB	BAS\$\$STOP_IO, R10	
					MOVL	FP, FMP	1320
					CLRL	OFFSET	1324
					MOVZBL	(AP), NUMARGS	1328
					MOVL	(AP)[NUMARGS], CHAN	1329
					MOVL	-4(AP)[NUMARGS], DECL	1330
					TSTL	CHAN	1336
					BGEQ	1\$	
					MOVZBL	#BAS\$K_ILLIO_CHA, -(SP)	
					CALLS	#1, BAS\$\$STOP	
					TSTL	CHAN	1338
					BNEQ	2\$	
					MNEGL	#7, LUN_NO	
					BRB	3\$	
					MOVL	CHAN, LUN_NO	
					MNEGL	#8, R0	1339
					JSB	BAS\$\$CB_PUSH	
					MOVL	12(FMP), -180(CCB)	1340
					BLBS	-4(CCB), 5\$	1342
					TSTL	LUN_NO	1344
					BGEQ	4\$	
					PUSHL	12(FMP)	1347
					CALLS	#1, BAS\$\$OPEN_ZERO	
					BRB	5\$	1344
					MOVZBL	#BAS\$K_IO_CHANOT, -(SP)	1351
					CALLS	#1, BAS\$\$STOP_IO	
					TSTL	-92(CCB)	1354
					BNEQ	6\$	
					MOVAB	BAS\$\$FIELD_KILL, -92(CCB)	
					MOVAB	BAS\$\$FIELD_KILL, R0	1356
					CMPL	-92(CCB), R0	
					BEQL	7\$	
					MOVZBL	#BAS\$K_ILLLACC, -(SP)	
					CALLS	#1, BAS\$\$STOP_IO	
					BISB2	#2, -1(CCB)	1358
					BISB2	#64, -95(CCB)	1360
					BLBC	-1(CCB), 8\$	1362

  

			OFFC	00000			
5A	00000000G	00	9E	00002			
55		5D	D0	00009			
		54	D4	0000C			
53		6C	9A	0000E			
56		6C43	D0	00011			
59	FC	AC43	D0	00015			
		56	D5	0001A			
		0B	18	0001C			
7E	00G	8F	9A	0001E			
00000000G	00	01	FB	00022			
		56	D5	00029	1\$:		
		05	12	0002B			
52		07	CE	0002D			
		03	11	00030			
52		56	D0	00032	2\$:		
50		08	CE	00035	3\$:		
	00000000G	00	16	00038			
FF4C	CB	0C	A5	D0	0003E		
17	FC	AB	E8	00044			
		52	D5	00048			
		0C	18	0004A			
		0C	A5	DD	0004C		
00000000G	00	01	FB	0004F			
		07	11	00056			
7E	00G	8F	9A	00058	4\$:		
6A		01	FB	0005C			
	A4	AB	D5	0005F	5\$:		
		06	12	00062			
A4	AB	0000V	CF	9E	00064		
50		0000V	CF	9E	0006A	6\$:	
50		A4	AB	D1	0006F		
		07	13	00073			
7E	00G	8F	9A	00075			
6A		01	FB	00079			
FF	AB	02	88	0007C	7\$:		
A1	AB	40	8F	88	00080		
07	FF	AB	E9	00085			

BASSRSTS\_FIELD  
1-023

D 2  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTSF1.B32:1

Page 10  
(4)

7E	00G	8F	9A	00089	MOVZBL	#BASSK_ILLILLACC, -(SP)	
6A		01	FB	0008D	CALLS	#1, BASS\$STOP_10	
58	EC	AB	DO	00090	8\$:	MOVL	-20(CCB), RBF
57	D2	AB	3C	00094	MOVZWL	-46(CCB), RSZ	1367
52		53	DO	00098	MOVL	NUMARGS, ARGNO	1368
		2D	11	0009B	BRB	11\$	1388
53		6C42	DO	0009D	9\$:	MOVL	(AP)[ARGNO], LEN
55	FC	AC42	DO	000A1	MOVL	-4(AP)[ARGNO], VAR	1383
54		53	C1	000A6	ADDL3	LEN, OFFSET, R0	1384
57		50	D1	000AA	CMPL	R0, RSZ	1386
		07	15	000AD	BLEQ	10\$	
7E	00G	8F	9A	000AF	MOVZBL	#BASSK_FIEOVEBUF, -(SP)	
6A		01	FB	000B3	CALLS	#1, BASS\$STOP_10	
	0120	8F	BB	000B6	10\$:	PUSHR	#M<R5,R8>
	0208	8F	BB	000BA	PUSHR	#M<R3,R9>	1388
		54	DD	000BE	PUSHL	OFFSET	
0000V	CF	56	DD	000C0	PUSHL	CHAN	
	54	06	FB	000C2	CALLS	#6, BASS\$FIELD_VAR	
	52	53	C0	000C7	ADDL2	LEN, OFFSET	1389
		02	C2	000CA	11\$:	SUBL2	#2, ARGNO
		CE	14	000CD	BGTR	9\$	1376
	00000000G	00	16	000CF	JSB	BASS\$CB_POP	1395
		04	000D5	RET			1397

; Routine Size: 214 bytes, Routine Base: \_BASS\$CODE + 0000

; 371 1398 1



```
373 1399 1 ROUTINE BASSFIELD_VAR (
374 1400 1     CHAN,
375 1401 1     OFFSET,
376 1402 1     LEN,
377 1403 1     DECL,
378 1404 1     VAR,
379 1405 1     RBF
380 1406 1 ) : CALL_CCB NOVALUE =
381 1407 1
382 1408 1
383 1409 1 **
384 1410 1 FUNCTIONAL DESCRIPTION:
385 1411 1     "Declares" a field variable. Such a variable refers to the
386 1412 1     buffer of an I/O channel. To avoid leaving obsolete addresses
387 1413 1     in a user's program each reference to a FIELD variable is
388 1414 1     interpreted. This routine puts the variable in the interpreter
389 1415 1     symbol table so it can be found by BASSFIELD_COPY.
390 1416 1
391 1417 1 FORMAL PARAMETERS:
392 1418 1
393 1419 1     CHAN.rl.v     An I/O channel. Need not be open yet.
394 1420 1     OFFSET.rl.v   The offset into that channel's buffer of the
395 1421 1                 start of the area referenced by the variable
396 1422 1     LEN.rl.v      The number of bytes referenced by the variable
397 1423 1     DECL.rl.v     An indication of the scope of the declaration
398 1424 1                 of the variable. This is a pointer to the major
399 1425 1                 frame (R11) if the variable is in the scope of
400 1426 1                 the major procedure, or a pointer to the minor
401 1427 1                 frame (R10) if the variable is in the scope of
402 1428 1                 a DEF.
403 1429 1     VAR.wt.d      The variable. Its storage is freed and it is
404 1430 1                 made to point to the buffer, so the compiled
405 1431 1                 code can do read accesses through it.
406 1432 1     RBF.ra.v      Address of the file's record buffer.
407 1433 1
408 1434 1 IMPLICIT INPUTS:
409 1435 1
410 1436 1     SYMSQ_ROOT.mq The queue of FIELD variables : the symbol table.
411 1437 1
412 1438 1 IMPLICIT OUTPUTS:
413 1439 1
414 1440 1     SYMSQ_ROOT.mq
415 1441 1
416 1442 1 ROUTINE VALUE:
417 1443 1 COMPLETION CODES:
418 1444 1
419 1445 1     NONE
420 1446 1
421 1447 1 SIDE EFFECTS:
422 1448 1
423 1449 1     Adds a symbol to the interpreter's symbol table, or modifies one
424 1450 1     already there.
425 1451 1
426 1452 1 --
427 1453 1
428 1454 2 BEGIN
429 1455 2
```

```

430      1456      2      EXTERNAL REGISTER
431      1457      CCB : REF BLOCK [, BYTE];
432      1458
433      1459      MAP
434      1460      VAR : REF BLOCK [8, BYTE];
435      1461
436      1462      LOCAL
437      1463      SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
438      1464      SEARCH_DONE;
439      1465
440      1466      +
441      1467      - If the symbol table root has not yet been initialized, initialize it.
442      1468
443      1469
444      1470      IF (.SYMSQ_ROOT [0] EQL 0)
445      1471      THEN
446      1472      BEGIN
447      1473
448      1474      LOCAL
449      1475      AST_STATUS;
450      1476
451      1477      AST_STATUS = $SETAST (ENBFLG = 0);
452      1478
453      1479      IF (.SYMSQ_ROOT [0] EQL 0)
454      1480      THEN
455      1481      BEGIN
456      1482      SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
457      1483      END;
458      1484
459      1485      IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
460      1486
461      1487      END;
462      1488
463      1489      +
464      1490      - Search the queue to see if this variable is already on it.
465      1491
466      1492      SYM = .SYMSQ_ROOT [0];
467      1493      SEARCH_DONE = 0;
468      1494
469      1495      DO
470      1496      BEGIN
471      1497
472      1498      IF (.SYM EQLA SYMSQ_ROOT)
473      1499      THEN
474      1500      SEARCH_DONE = 1
475      1501      ELSE
476      1502
477      1503      IF (.SYM [SYMSA_VAR] EQLA .VAR)
478      1504      THEN
479      1505      BEGIN
480      1506
481      1507      IF (.SYM [SYMSV_INVALID]) THEN BASS$STOP_IO (BASSK_ILLFIEVAR);
482      1508
483      1509      SEARCH_DONE = 3;
484      1510      END;
485      1511
486      1512      IF ( NOT .SEARCH_DONE) THEN SYM = .SYM [SYMSA_NEXT];
```



```
487 1513
488 1514
489 1515      END
490 1516      UNTIL (.SEARCH_DONE);
491 1517      IF (.SEARCH_DONE EQL 1)
492 1518      THEN
493 1519      BEGIN
494 1520
495 1521      + We must create a symbol table entry.
496 1522
497 1523
498 1524      BUILTIN
499 1525      INSQUE;
500 1526
501 1527      LOCAL
502 1528      GET_VM_STATUS,
503 1529      INSQUE_ADDR;
504 1530
505 1531      GET_VM_STATUS = LIB$GET_VM (%REF (SYM$K_LENGTH), SYM);
506 1532
507 1533      IF ( NOT .GET_VM_STATUS) THEN BAS$$STOP_10 (BAS$K_MAXMEMEXC);
508 1534
509 1535      INSQUE_ADDR = SYM$Q_ROOT [1];
510 1536      INSQUE (.SYM, ..INSQUE_ADDR);          ! Tail of queue
511 1537
512 1538      + Make sure the string is empty.
513 1539
514 1540      STR$FREE1_DX (.VAR);
515 1541      END;
516 1542
517 1543      + Fill in the symbol table entry
518 1544
519 1545
520 1546      SYM [SYM$S_CHAN] = .CHAN;
521 1547      SYM [SYM$S_OFFSET] = .OFFSET;
522 1548      SYM [SYM$S_LEN] = .LEN;
523 1549      SYM [SYM$S_DECL] = .DECL;
524 1550      SYM [SYM$S_VAR] = .VAR;
525 1551      SYM [SYM$S_INVALID] = 0;
526 1552      VAR [DSC$W_LENGTH] = MAX (0, .LEN);
527 1553      VAR [DSC$B_CLASS] = DSC$K_CLASS_S;
528 1554      VAR [DSC$A_POINTER] = .RBF + .OFFSET;
529 1555      RETURN;
530 1556      END;                                     ! end of BAS$FIELD_VAR
```

.EXTRN SYS\$SETAST

```
007C 00000 BAS$FIELD VAR:
56 00000000G 00 9E 00002  .WORD      Save R2,R3,R4,R5,R6
55 00000000G 00 9E 00009  MOVAB     BAS$$STOP_10, R6
54 00000000G EF 9E 00010  MOVAB     SYS$SETAST, R5
5E          08 C2 00017  MOVAB     SYM$Q_ROOT, R4
          64 D5 0001A  SUBL2     #8, SP
          1D 12 0001C  TSTL     SYM$Q_ROOT
          BNEQ     Z8
```

1399

1470

			7E	D4	0001E	CLRL	-(SP)	1477	
	65		01	FB	00020	CALLS	#1, SYSSSETAST		
			64	D5	00023	TSTL	SYMSQ_ROOT	1479	
			0A	12	00025	BNEQ	18		
	51		64	9E	00027	MOVAB	SYMSQ_ROOT, R1	1482	
04	A4		51	D0	0002A	MOVL	R1, SYMSQ_ROOT+4		
	64		51	D0	0002E	MOVL	R1, SYMSQ_ROOT		
	09		50	D1	00031	18:	CMPL	AST_STATUS, #9	1485
			05	12	00034	BNEQ	28		
			01	DD	00036	PUSHL	#1		
	65		01	FB	00038	CALLS	#1, SYSSSETAST		
04	AE		64	D0	0003B	28:	MOVL	SYMSQ_ROOT, SYM	1492
			53	D4	0003F	CLRL	SEARCH_DONE	1493	
	52	04	AE	D0	00041	38:	MOVL	SYM, R2	1498
	50		64	9E	00045	MOVAB	SYMSQ_ROOT, R0		
	50		52	D1	00048	CMPL	R2, R0		
			05	12	00048	BNEQ	48		
	53		01	D0	0004D	MOVL	#1, SEARCH_DONE	1500	
			15	11	00050	BRB	68		
14	AC	18	A2	D1	00052	48:	CMPL	24(R2), VAR	1503
			0E	12	00057	BNEQ	68		
	07	1C	A2	E9	00059	BLBC	28(R2), 58	1507	
	7E	00G	8F	9A	0005D	MOVZBL	#BASBK_ILLFIEVAR, -(SP)		
	66		01	FB	00061	CALLS	#1, BAS\$\$STOP_10		
	53		03	D0	00064	58:	MOVL	#3, SEARCH_DONE	1509
	07		53	E8	00067	68:	BLBS	SEARCH_DONE, 78	1512
04	AE		62	D0	0006A	MOVL	(R2), SYM		
	D0		53	E9	0006E	BLBC	SEARCH_DONE, 38	1515	
	01		53	D1	00071	78:	CMPL	SEARCH_DONE, #1	1517
			2E	12	00074	BNEQ	98		
		04	AE	9F	00076	PUSHAB	SYM	1531	
04	AE		20	D0	00079	MOVL	#32, 4(SP)		
		04	AE	9F	0007D	PUSHAB	4(SP)		
00000000G	00		02	FB	00080	CALLS	#2, LIB\$GET_VM		
	07		50	E8	00087	BLBS	GET_VM_STATUS, 88	1533	
	7E	00G	8F	9A	0008A	MOVZBL	#BASBK_MAXMEMEXC, -(SP)		
	66		01	FB	0008E	CALLS	#1, BAS\$\$STOP_10		
	50	04	A4	9E	00091	88:	MOVAB	SYMSQ_ROOT+4, INSQUE_ADDR	1535
0C	B0	04	BE	0E	00095	INSQUE	@SYM, 20(INSQUE_ADDR)	1536	
		14	AC	DD	0009A	PUSHL	VAR	1540	
00000000G	00		01	FB	0009D	CALLS	#1, STR\$FREE1_DX		
	50	04	AE	D0	000A4	98:	MOVL	SYM, R0	1546
08	A0	04	AC	7D	000AB	MOVQ	CHAN, 8(R0)		
10	A0	0C	AC	7D	000AD	MOVQ	LEN, 16(R0)	1548	
	51	14	AC	D0	000B2	MOVL	VAR, R1	1550	
18	A0		51	D0	000B6	MOVL	R1, 24(R0)		
1C	A0		01	8A	000BA	BICB2	#1, 28(R0)	1551	
	50	0C	AC	D0	000BE	MOVL	LEN, R0	1552	
			02	18	000C2	BGEQ	108		
			50	D4	000C4	CLRL	R0		
	61		50	B0	000C6	108:	MOVW	R0, (R1)	
03	A1		01	90	000C9	MOVB	#1, 3(R1)	1553	
18	AC	08	AC	C1	000CD	ADDL3	OFFSET, RBF, 4(R1)	1554	
			04	000D4	RET			1556	



BASRSTS\_FIELD  
1-023

1 2  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 BLISS-32 V4.0-742  
[BASRTL.SRC]BASRSTSF1.B32:1

Page 15  
(5)

: 531 1557 1

```
533 1558 1 GLOBAL ROUTINE BASSFIELD_CLEAR (      ! Undeclare a field variable
534 1559 1     VAR                                ! The variable being cleared
535 1560 1     ) : NOVALUE =
536 1561 1
537 1562 1
538 1563 1 ++
539 1564 1 FUNCTIONAL DESCRIPTION:
540 1565 1     Undeclare a possible FIELD variable. This routine is called
541 1566 1     prior to any BASIC statement that causes a field variable to
542 1567 1     lose its FIELD attribute, if that variable has a FIELD
543 1568 1     statement associated with it anywhere in the program.
544 1569 1
545 1570 1 FORMAL PARAMETERS:
546 1571 1
547 1572 1     VAR.at.d      The variable. Only the address of its
548 1573 1                  descriptor is used, to scan the symbol table.
549 1574 1
550 1575 1 IMPLICIT INPUTS:
551 1576 1
552 1577 1     SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.
553 1578 1
554 1579 1 IMPLICIT OUTPUTS:
555 1580 1
556 1581 1     SYMSQ_ROOT.mq
557 1582 1
558 1583 1 ROUTINE VALUE:
559 1584 1 COMPLETION CODES:
560 1585 1
561 1586 1     NONE
562 1587 1
563 1588 1 SIDE EFFECTS:
564 1589 1
565 1590 1     May remove a symbol from the symbol table.
566 1591 1
567 1592 1 --
568 1593 1
569 1594 2 BEGIN
570 1595 2
571 1596 2 MAP
572 1597 2     VAR : REF BLOCK [8, BYTE];
573 1598 2
574 1599 2 LOCAL
575 1600 2     SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
576 1601 2     SEARCH_DONE;
577 1602 2
578 1603 2 ++
579 1604 2     If the symbol table root has not yet been initialized, initialize it.
580 1605 2
581 1606 2
582 1607 2 IF (.SYMSQ_ROOT [0] EQL 0)
583 1608 2 THEN
584 1609 2     BEGIN
585 1610 2
586 1611 2     LOCAL
587 1612 2     AST_STATUS;
588 1613 2
589 1614 2     AST_STATUS = $SETAST (ENBFLG = 0);
```

```
590 1615 3
591 1616
592 1617 IF (.SYMSQ_ROOT [0] EQL 0)
593 1618 THEN
594 1619 BEGIN
595 1620 SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
596 1621 END;
597 1622 IF (.AST_STATUS EQL SSS_WASSET) THEN $SETAST (ENBFLG = 1);
598 1623
599 1624 END;
600 1625
601 1626
602 1627 + Search the symbol table, removing this variable if it is present.
603 1628 -
604 1629 SYM = .SYMSQ_ROOT [0];
605 1630 SEARCH_DONE = 0;
606 1631
607 1632 DO
608 1633 BEGIN
609 1634
610 1635 IF (.SYM EQLA SYMSQ_ROOT)
611 1636 THEN
612 1637 SEARCH_DONE = 1
613 1638 ELSE
614 1639
615 1640 IF (.SYM [SYMSA_VAR] EQL .VAR)
616 1641 THEN
617 1642 BEGIN
618 1643 + We must delete this symbol from the symbol table.
619 1644 -
620 1645
621 1646 BUILTIN
622 1647 REMQUE;
623 1648
624 1649 LOCAL
625 1650 FREE_VM_STATUS,
626 1651 TEMP;
627 1652
628 1653 IF (.SYM [SYMSV_INVALID]) THEN BAS$$STOP (BAS$K_ILLFIEVAR);
629 1654
630 1655 REMQUE (.SYM, TEMP);
631 1656 VAR [DSC$W_LENGTH] = 0;
632 1657 VAR [DSC$B_CLASS] = DSC$K_CLASS_D;
633 1658 VAR [DSC$A_POINTER] = 0;
634 1659 FREE_VM_STATUS = LIB$FREE_VM (%REF (SYM$K_LENGTH), TEMP);
635 1660
636 1661 IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP (BAS$K_PROLOSSOR);
637 1662
638 1663 SEARCH_DONE = 1
639 1664 END
640 1665 ELSE
641 1666 SYM = .SYM [SYMSA_NEXT];
642 1667
643 1668
644 1669 END
645 1670 UNTIL (.SEARCH_DONE);
646 1671 2
```



BASRSTS\_FIELD  
1-023

L 2  
16-Sep-1984 01:07:30 VAX-11 B11sg-32 V4.0-742  
14-Sep-1984 11:56:38 [BASRTL.SRC]BASRSTSFI.B32:1

Page 18  
(6)

: 647 1672 1 END:

: end of BASFIELD\_CLEAR

			007C 00000	.ENTRY	BASFIELD_CLEAR, Save R2,R3,R4,R5,R6	1558
56	00000000G	00	9E 00002	MOVAB	BAS\$\$STOP, R6	
55	00000000G	00	9E 00009	MOVAB	SYSS\$SETAST, R5	
54	00000000G	EF	9E 00010	MOVAB	SYMSQ_ROOT, R4	
5E		08	C2 00017	SUBL2	#8, SP	
		64	D5 0001A	TSTL	SYMSQ_ROOT	1607
		1D	12 0001C	BNEQ	2\$	
		7E	D4 0001E	CLRL	-(SP)	1614
65		01	FB 00020	CALLS	#1, SYSS\$SETAST	
		64	D5 00023	TSTL	SYMSQ_ROOT	1616
		0A	12 00025	BNEQ	1\$	
51		64	9E 00027	MOVAB	SYMSQ_ROOT, R1	1619
04	A4	51	D0 0002A	MOVL	R1, SYMSQ_ROOT+4	
64		51	D0 0002E	MOVL	R1, SYMSQ_ROOT	
09		50	D1 00031	CMPL	AST_STATUS, #9	1622
		05	12 00034	BNEQ	2\$	
		01	DD 00036	PUSHL	#1	
65		01	FB 00038	CALLS	#1, SYSS\$SETAST	
52		64	D0 0003B	MOVL	SYMSQ_ROOT, SYM	1629
		53	D4 0003E	CLRL	SEARCH_DONE	1630
50		64	9E 00040	MOVAB	SYMSQ_ROOT, R0	1635
50		52	D1 00043	CMPL	SYM, R0	
		3E	13 00046	BEQL	5\$	
04	AC	18	A2 D1 00048	CMPL	24(SYM), VAR	1640
		3C	12 0004D	BNEQ	6\$	
07		1C	A2 E9 0004F	BLBC	28(SYM), 4\$	1654
7E		00G	8F 9A 00053	MOVZBL	#BAS\$K_ILLFIEVAR, -(SP)	
66		01	FB 00057	CALLS	#1, BAS\$\$STOP	
04	AE	62	0F 0005A	REMQUE	(SYM), TEMP	1656
50		04	AC D0 0005E	MOVL	VAR, R0	1657
		60	B4 00062	CLRW	(R0)	
03	A0	02	90 00064	MOVB	#2, 3(R0)	1658
		04	A0 D4 00068	CLRL	4(R0)	1659
		04	AE 9F 0006B	PUSHAB	TEMP	1660
04	AE	20	D0 0006E	MOVL	#32, 4(SP)	
		04	AE 9F 00072	PUSHAB	4(SP)	
00000000G	00	02	FB 00075	CALLS	#2, LIB\$FREE_VM	
07		50	E8 0007C	BLBS	FREE_VM_STATUS, 5\$	1662
7E		00G	8F 9A 0007F	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	
66		01	FB 00083	CALLS	#1, BAS\$\$STOP	
53		01	D0 00086	MOVL	#1, SEARCH_DONE	1664
		03	11 00089	BRB	7\$	
52		62	D0 0008B	MOVL	(SYM), SYM	1667
AF		53	E9 0008E	BLBC	SEARCH_DONE, 3\$	1670
		04	00091	RET		1672

: Routine Size: 146 bytes, Routine Base: \_BAS\$CODE + 01AB

: 648 1673 1

```
650 1674 1 GLOBAL ROUTINE BASSFIELD_COPY (      | Copy to or from a FIELD variable
651 1675 1     STMT_TYPE,                        | Either LSET or RSET
652 1676 1     VAR2,                          | The destination variable
653 1677 1     VAR1,                          | The source variable
654 1678 1     ) : NOVALUE =
655 1679 1
656 1680 1 ++
657 1681 1 FUNCTIONAL DESCRIPTION:
658 1682 1
659 1683 1     Copies between two string variables. One or the other may
660 1684 1     by FIELD variables, but not both. Because the compiler cannot
661 1685 1     be sure if a FIELD statement has been issued to a variable
662 1686 1     (since it cannot trace program flow) it is possible that
663 1687 1     neither variable is FIELD.
664 1688 1
665 1689 1 FORMAL PARAMETERS:
666 1690 1
667 1691 1     STMT_TYPE.rl.v 0 = this is an LSET statement, 1 = RSET
668 1692 1     VAR2.wt.dx    The destination of the copy. This may be a
669 1693 1                   FIELD variable.
670 1694 1     VAR1.rt.dx    The source for the copy. This may be a FIELD
671 1695 1                   variable.
672 1696 1 IMPLICIT INPUTS:
673 1697 1
674 1698 1     SYMSQ_ROOT.rq The queue of FIELD variables : the symbol table.
675 1699 1
676 1700 1 IMPLICIT OUTPUTS:
677 1701 1
678 1702 1     NONE
679 1703 1
680 1704 1 ROUTINE VALUE:
681 1705 1 COMPLETION CODES:
682 1706 1
683 1707 1     NONE
684 1708 1
685 1709 1 SIDE EFFECTS:
686 1710 1
687 1711 1     May write into or read from an I/O buffer.
688 1712 1
689 1713 1 --
690 1714 1
691 1715 2 BEGIN
692 1716 2
693 1717 2 BUILTIN
694 1718 2     FP;
695 1719 2
696 1720 2 GLOBAL REGISTER
697 1721 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
698 1722 2
699 1723 2 MAP
700 1724 2     VAR1 : REF BLOCK [8, BYTE],
701 1725 2     VAR2 : REF BLOCK [8, BYTE];
702 1726 2
703 1727 2 LOCAL
704 1728 2     FMP : REF BLOCK [, BYTE],
705 1729 2     SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
706 1730 2     SEARCH_DONE.
```

```

707 1731 2      VAR1_DESC : BLOCK [8, BYTE],
708 1732      VAR2_DESC : BLOCK [8, BYTE],
709 1733      VAR1_DESC_ADR : REF BLOCK [8, BYTE],
710 1734      VAR2_DESC_ADR : REF BLOCK [8, BYTE],
711 1735      VAR1_CHAN;
712 1736      VAR2_CHAN;
713 1737
714 1738      FMP = .FP;
715 1739
716 1740      + If the symbol table root has not yet been initialized, initialize it.
717 1741      -
718 1742
719 1743      IF (.SYMSQ_ROOT [0] EQL 0)
720 1744      THEN
721 1745          BEGIN
722 1746              LOCAL
723 1747                  AST_STATUS;
724 1748                  AST_STATUS = $SETAST (ENBFLG = 0);
725 1749
726 1750              IF (.SYMSQ_ROOT [0] EQL 0)
727 1751              THEN
728 1752                  BEGIN
729 1753                      SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
730 1754                  END;
731 1755
732 1756              IF (.AST_STATUS EQL $$$_WASSET) THEN $SETAST (ENBFLG = 1);
733 1757
734 1758              END;
735 1759
736 1760
737 1761      + Search the queue to see if the input variable is on it.
738 1762      -
739 1763      SYM = .SYMSQ_ROOT [0];
740 1764      SEARCH_DONE = 0;
741 1765
742 1766      DO
743 1767          BEGIN
744 1768              IF (.SYM EQLA SYMSQ_ROOT)
745 1769              THEN
746 1770                  SEARCH_DONE = 1
747 1771              ELSE
748 1772                  IF (.SYM [SYMSA_VAR] EQLA .VAR1) THEN SEARCH_DONE = 3;
749 1773                  IF ( NOT .SEARCH_DONE) THEN SYM = .SYM [SYMSA_NEXT];
750 1774
751 1775                  END
752 1776              UNTIL (.SEARCH_DONE);
753 1777
754 1778              IF (.SEARCH_DONE EQL 1)
755 1779              THEN
756 1780                  BEGIN
757 1781
758 1782
759 1783
760 1784
761 1785
762 1786
763 1787      + The variable is not in the symbol table. That must mean that it
```



```
.. 764 1788 3 ! is not a FIELD variable.
765 1789 !-
766 1790     VAR1_DESC_ADR = .VAR1;
767 1791     VAR1_CHAN = 0;
768 1792     END
769 1793     ELSE
770 1794     BEGIN
771 1795     !+
772 1796     !- Don't touch a variable marked invalid.
773 1797     !-
774 1798     IF (.SYM [SYMSV_INVALID]) THEN BAS$$STOP (BAS$K_ILLFIEVAR);
775 1799
776 1800     !+
777 1801     !- The variable is in the symbol table. Construct a descriptor for it.
778 1802
779 1803     VAR1_DESC_ADR = VAR1_DESC;
780 1804     VAR1_DESC [DSC$W_LENGTH] = MAX (0, .SYM [SYMSL_LEN]);
781 1805     VAR1_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
782 1806     VAR1_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
783 1807     VAR1_CHAN = .SYM [SYMSL_CHAN];
784 1808
785 1809     IF (.VAR1_CHAN EQL 0) THEN VAR1_CHAN = LUB$K_LUN_INPU;
786 1810
787 1811     BAS$$CB_PUSH (.VAR1_CHAN, LUB$K_LUN_INPU);
788 1812     CCB [ISB$A_USER_FP] = .FMP [SF$C_SAVE_FP];
789 1813     VAR1_DESC [DSC$A_POINTER] = .CCB [LUB$A_RBUF_ADR] + .SYM [SYMSL_OFFSET];
790 1814
791 1815     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_IO_CHANOT);
792 1816
793 1817     IF (.CCB [LUB$W_RBUF_SIZE] LSSU .SYM [SYMSL_OFFSET] + .SYM [SYMSL_LEN])
794 1818     THEN
795 1819     BAS$$STOP_IO (BAS$K_FIEOVEBUF);
796 1820
797 1821     END;
798 1822
799 1823     !+
800 1824     !- Search the queue to see if the output variable is on it.
801 1825
802 1826     SYM = .SYMSQ_ROOT [0];
803 1827     SEARCH_DONE = 0;
804 1828
805 1829     DO
806 1830     BEGIN
807 1831
808 1832     IF (.SYM EQLA SYMSQ_ROOT)
809 1833     THEN
810 1834     SEARCH_DONE = 1
811 1835     ELSE
812 1836
813 1837     IF (.SYM [SYM$A_VAR] EQLA .VAR2) THEN SEARCH_DONE = 3;
814 1838
815 1839     IF ( NOT .SEARCH_DONE) THEN SYM = .SYM [SYM$A_NEXT];
816 1840
817 1841     END
818 1842     UNTIL (.SEARCH_DONE);
819 1843
820 1844
```

```
1845 IF (.SEARCH_DONE EQL 1)
1846 THEN
1847 BEGIN
1848
1849 + The variable is not in the symbol table. That must mean that it
1850 is not a FIELD variable.
1851
1852 VAR2_DESC_ADR = .VAR2;
1853 VAR2_CHAN = 0;
1854 END
1855 ELSE
1856 BEGIN
1857
1858 IF (.SYM [SYMSV_INVALID]) THEN BAS$$STOP (BAS$K_ILLFIEVAR);
1859
1860 + The variable is in the symbol table. Construct a descriptor for it.
1861
1862 VAR2_DESC_ADR = VAR2_DESC;
1863 VAR2_DESC [DSC$W_LENGTH] = MAX (0, .SYM [SYMSL_LEN]);
1864 VAR2_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1865 VAR2_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
1866 VAR2_CHAN = .SYM [SYMSL_CHAN];
1867
1868 IF (.VAR2_CHAN EQL 0) THEN VAR2_CHAN = LUB$K_LUN_INPU;
1869
1870 BAS$$CB_PUSH (.VAR2_CHAN, LUB$K_LUN_INPU);
1871 CCB [ISB$A_USER_FP] = .FMP [SF$[SAVE_FP]];
1872 VAR2_DESC [DSC$A_POINTER] = .CCB [LUB$A_RBUF_ADR] + .SYM [SYMSL_OFFSET];
1873
1874 IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_IO_CHANOT);
1875
1876 IF (.CCB [LUB$W_RBUF_SIZE] LSSU .SYM [SYMSL_OFFSET] + .SYM [SYMSL_LEN])
1877 THEN
1878 BAS$$STOP_IO (BAS$K_FIEOVEBUF);
1879
1880 END;
1881
1882 + Copy from the input variable to the output variable.
1883 We must observe the semantics of the statement type.
1884
1885 CASE .STMT_TYPE FROM STMT_TYPE_LSET TO STMT_TYPE_RSET OF
1886 SET
1887 [STMT_TYPE_LSET] :
1888 STR$COPY_DX (.VAR2_DESC_ADR, .VAR1_DESC_ADR);
1889
1890 [STMT_TYPE_RSET] :
1891 BAS$RSET (.VAR2_DESC_ADR, .VAR1_DESC_ADR);
1892
1893 TES;
1894
1895 + Release register CCB
1896
1897
1898
1899
1900
1901
```

```
878 1902 3 IF (.VAR2_CHAN NEQ 0)
879 1903 THEN
880 1904 BEGIN
881 1905 BAS$$CB_GET ();
882 1906 BAS$$CB_POP ();
883 1907 END;
884 1908
885 1909 IF (.VAR1_CHAN NEQ 0)
886 1910 THEN
887 1911 BEGIN
888 1912 BAS$$CB_GET ();
889 1913 BAS$$CB_POP ();
890 1914 END;
891 1915
892 1916 END;
```

! end of BAS\$FIELD\_COPY

		OFFC	00000		.ENTRY	BAS\$FIELD_COPY, Save R2,R3,R4,R5,R6,R7,R8,-		
	5A	00000000G	00	9E	00002	MOVAB	R9,R10,R11	1674
	59	00000000G	00	9E	00009	MOVAB	SY\$\$SETAST, R10	
	58	00000000'	EF	9E	00010	MOVAB	BAS\$\$STOP, R9	
	5E		10	C2	00017	MOVAB	SYMSQ_ROOT, R8	
	55		5D	D0	0001A	SUBL2	#16, SP	1738
			68	D5	0001D	MOVL	FP, FMP	1743
			1D	12	0001F	TSTL	SYMSQ_ROOT	
			7E	D4	00021	BNEQ	2\$	
	6A		01	FB	00023	CLRL	-(SP)	1750
			68	D5	00026	CALLS	#1, SY\$\$SETAST	
			0A	12	00028	TSTL	SYMSQ_ROOT	1752
	51		68	9E	0002A	BNEQ	1\$	
04	A8		51	D0	0002D	MOVAB	SYMSQ_ROOT, R1	1755
	68		51	D0	00031	MOVL	R1, SYMSQ_ROOT+4	
	09		50	D1	00034	MOVL	R1, SYMSQ_ROOT	
			05	12	00037	CMPL	AST_STATUS, #9	1758
			01	DD	00039	BNEQ	2\$	
	6A		01	FB	0003B	PUSHL	#1	
	53		68	D0	0003E	CALLS	#1, SY\$\$SETAST	
			54	D4	00041	MOVL	SYMSQ_ROOT, SYM	1765
	50		68	9E	00043	CLRL	SEARCH_DONE	1766
	50		53	D1	00046	MOVAB	SYMSQ_ROOT, R0	1771
			05	12	00049	CMPL	SYM, R0	
	54		01	D0	0004B	BNEQ	4\$	
			0A	11	0004E	MOVL	#1, SEARCH_DONE	1773
0C	AC	18	A3	D1	00050	BRB	5\$	
			03	12	00055	CMPL	24(SYM), VAR1	1776
	54		03	D0	00057	BNEQ	5\$	
	06		54	E8	0005A	MOVL	#3, SEARCH_DONE	
	53		63	D0	0005D	BLBS	SEARCH_DONE, 6\$	1778
	E0		54	E9	00060	MOVL	(SYM), -SYM	
	01		54	D1	00063	BLBC	SEARCH_DONE, 3\$	1781
			08	12	00066	CMPL	SEARCH_DONE, #1	1783
	57	0C	AC	D0	00068	BNEQ	7\$	
			56	D4	0006C	MOVL	VAR1, VAR1_DESC_ADR	1790
						CLRL	VAR1_CHAN	1791



					67	11	0006E		BRB	12\$	1783	
					A3	E9	00070	7\$:	BLBC	28(SYM), 8\$	1799	
					8F	9A	00074		MOVZBL	#BAS\$K ILLFIEVAR, -(SP)		
					01	FB	00078		CALLS	#1, BAS\$STOP		
					AE	9E	0007B	8\$:	MOVAB	VAR1_DESC, VAR1_DESC_ADR	1804	
					A3	D0	0007F		MOVL	16(SYM), R0	1805	
					02	18	00083		BGEQ	9\$		
					50	D4	00085		CLRL	R0		
					50	B0	00087	9\$:	MOVW	R0, VAR1_DESC		
					8F	B0	0008B		MOVW	#270, VAR1_DESC+2	1806	
					A3	D0	00091		MOVL	8(SYM), VAR1_CHAN	1808	
					03	12	00095		BNEQ	10\$	1810	
					07	CE	00097		MNEGL	#7, VAR1_CHAN		
					07	CE	0009A	10\$:	MNEGL	#7, R0	1812	
					56	D0	0009D		MOVL	VAR1_CHAN, R2		
					52	00	000A0		JSB	BAS\$CB_PUSH		
					00	A5	000A6		MOVL	12(FMP), -180(CCB)	1813	
					OC	A3	000AC		ADDL3	12(SYM), -20(CCB), VAR1_DESC+4	1814	
					OC	AB	000B3		BLBS	-4(CCB), 11\$	1816	
					07	8F	9A	000B7	MOVZBL	#BAS\$K IO_CHANOT, -(SP)		
					69	01	FB	000BB	CALLS	#1, BAS\$STOP		
					A3	C1	000BE	11\$:	ADDL3	16(SYM), 12(SYM), R0	1818	
					10	00	ED	000C4	CMPZV	#0, #16, -46(CCB), R0		
					7E	0B	1E	000CA	BGEQU	12\$		
					00	8F	9A	000CC	MOVZBL	#BAS\$K FIEOVEBUF, -(SP)	1820	
					53	01	FB	000D0	CALLS	#1, BAS\$STOP IO		
					50	68	D0	000D7	MOVL	SYM\$0 ROOT, SYM	1827	
					50	54	D4	000DA	CLRL	SEARCH_DONE	1828	
					50	68	9E	000DC	MOVAB	SYM\$0 ROOT, R0	1833	
					53	D1	000DF		CMP	SYM, R0		
					05	12	000E2		BNEQ	14\$		
					54	01	D0	000E4	MOVL	#1, SEARCH_DONE	1835	
					0A	11	000E7		BRB	15\$		
					08	A3	D1	000E9	14\$:	CMP	24(SYM), VAR2	1838
					03	12	000EE		BNEQ	15\$		
					54	03	D0	000F0	MOVL	#3, SEARCH_DONE		
					06	54	E8	000F3	15\$:	BLBS	SEARCH_DONE, 16\$	1840
					53	63	D0	000F6	MOVL	(SYM), SYM		
					E0	54	E9	000F9	BLBC	SEARCH_DONE, 13\$	1843	
					01	54	D1	000FC	16\$:	CMP	SEARCH_DONE, #1	1845
					54	08	12	000FF	BNEQ	17\$		
					07	AC	D0	00101	MOVL	VAR2, VAR2_DESC_ADR	1852	
					7E	52	D4	00105	CLRL	VAR2_CHAN	1853	
					69	62	11	00107	BRB	22\$	1845	
					54	A3	E9	00109	17\$:	BLBC	28(SYM), 18\$	1858
					50	8F	9A	0010D	MOVZBL	#BAS\$K ILLFIEVAR, -(SP)		
					50	01	FB	00111	CALLS	#1, BAS\$STOP		
					6E	9E	00114	18\$:	MOVAB	VAR2_DESC, VAR2_DESC_ADR	1863	
					50	A3	D0	00117	MOVL	16(SYM), R0	1864	
					02	18	0011B		BGEQ	19\$		
					50	D4	0011D		CLRL	R0		
					6E	50	B0	0011F	19\$:	MOVW	R0, VAR2_DESC	
					02	8F	B0	00122	MOVW	#270, VAR2_DESC+2	1865	
					52	A3	D0	00128	MOVL	8(SYM), VAR2_CHAN	1867	
					52	03	12	0012C	BNEQ	20\$	1869	
					50	07	CE	0012E	MNEGL	#7, VAR2_CHAN		
					50	07	CE	00131	20\$:	MNEGL	#7, R0	1871

BASRSTS\_FIELD  
1-023

F 3  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 BLISS-32 V4.0-742  
[BASRTL.SRC]BASRSTSF1.B32:1

Page 25  
(7)

04	AE	FF4C EC	CB AB 07 7E 69 A3 10	00000000G OC OC FC 00G 10 00 00G 00 00 0011	00 A5 A3 AB 8F 01 A3 00 08 8F 01 AC 0004	16 D0 C1 E8 9A FB C1 ED 1E 9A FB CF 00170	00134 0013A 00140 00147 0014B 0014F 00152 00158 0015E 00160 00164 0016B 00170	218: 218: 228: 238:	JSB MOVL ADDL3 BLBS MOVZBL CALLS ADDL3 CMPZV BGEQU MOVZBL CALLS CASEL .WORD PUSHR CALLS BRB PUSHR CALLS TSTL BEQL JSB JSB TSTL BEQL JSB JSB RET	BAS\$\$CB_PUSH 12(FMP), -180(CCB) 12(SYM), -20(CCB), VAR2_DESC+4 -4(CCB), 218 #BAS\$K_IO_CHANOT, -(SP) #1, BAS\$\$STOP 16(SYM), 12(SYM), R3 #0, #16, -46(CCB), R3 228 #BAS\$K_FIEOVEBUF, -(SP) #1, BAS\$\$STOP_IO STMT_TYPE, #0, #1 248-238, - 258-238 #M<R4, R7> #2, STR\$COPY_DX 268 #M<R4, R7> #2, BAS\$RSET VAR2_CHAN 278 BAS\$\$CB_GET BAS\$\$CB_POP VAR1_CHAN 288 BAS\$\$CB_GET BAS\$\$CB_POP	1872 1873 1875 1877 1879 1888 1892 1895 1902 1905 1906 1909 1912 1913 1916
----	----	------------	--	---	--	---	---	------------------------------	---	---	--

; Routine Size: 429 bytes, Routine Base: \_BAS\$CODE + 023D

; 893 1917 1

```

895 1918 1 GLOBAL ROUTINE BASSFIELD_COP_R (
896 1919 1     STMT_TYPE,
897 1920 1     VAR2,
898 1921 1     VAR1_LEN,
899 1922 1     VAR1_ADDR
900 1923 1 ) : NOVALUE =
901 1924 1
902 1925 1
903 1926 1
904 1927 1
905 1928 1
906 1929 1
907 1930 1
908 1931 1
909 1932 1
910 1933 1
911 1934 1
912 1935 1
913 1936 1
914 1937 1
915 1938 1
916 1939 1
917 1940 1
918 1941 1
919 1942 1
920 1943 1
921 1944 1
922 1945 1
923 1946 1
924 1947 1
925 1948 1
926 1949 1
927 1950 1
928 1951 1
929 1952 1
930 1953 1
931 1954 1
932 1955 1
933 1956 1
934 1957 1
935 1958 1
936 1959 1
937 1960 1
938 1961 1
939 1962 1
940 1963 1
941 1964 1
942 1965 1
943 1966 1
944 1967 1
945 1968 1
946 1969 1
947 1970 1
948 1971 1
949 1972 1

GLOBAL ROUTINE BASSFIELD_COP_R (
    STMT_TYPE,
    VAR2,
    VAR1_LEN,
    VAR1_ADDR
) : NOVALUE =

**
FUNCTIONAL DESCRIPTION:

    This is an alternate entry point for BASSFIELD_COPY, which the
    compiler uses to avoid having to build a descriptor for a string
    constant. This code builds the descriptor and calls BASSFIELD_COPY.

FORMAL PARAMETERS:

    STMT_TYPE.rl.v  0 = this is an LSET statement, 1 = RSET
    VAR2.wt.dx      The destination of the copy. This may be a
                    FIELD variable.
    VAR1_LEN.rl.v   The number of bytes in the source
    VAR1_ADDR.rt.r  The address of the source

IMPLICIT INPUTS:

    SYMSQ_ROOT.rq  The queue of FIELD variables : the symbol table.

IMPLICIT OUTPUTS:

    NONE

ROUTINE VALUE:

COMPLETION CODES:

    NONE

SIDE EFFECTS:

    May write into or read from an I/O buffer.

--
BEGIN
LOCAL
    VAR1 : BLOCK [8, BYTE];
    VAR1 [DSCSW_LENGTH] = .VAR1_LEN;
    VAR1 [DSCSB_DTYPE] = DSCSK_DTYPE_T;
    VAR1 [DSCSB_CLASS] = DSCSK_CLASS_S;
    VAR1 [DSCSA_POINTER] = .VAR1_ADDR;

    Now do the copy.

    BASSFIELD_COPY (.STMT_TYPE, .VAR2, VAR1);
END;
! Build source descriptor here
! end of BASSFIELD_COP_R

```



BASRSTS\_FIELD  
1-023

M 3  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTS.F1.832;1

Page 27  
(8)

	5E		0000	00000
	6E	0C	08	C2 00002
02	AE	010E	AC	B0 00005
04	AE	10	8F	B0 00009
			AC	D0 0000F
	7E	04	5E	DD 00014
FE34	CF		AC	7D 00016
			03	FB 0001A
			04	0001F

```
.ENTRY BASFIELD_COPY_R, Save nothing
SUBL2 #8, SP
MOVW VAR1_LEN, VAR1
MOVW #270, VAR1+2
MOVL VAR1_ADDR, VAR1+4
PUSHL SP
MOVQ STMT_TYPE, -(SP)
CALLS #3, BASFIELD_COPY
RET
```

```
: 1918
: 1964
: 1965
: 1967
: 1971
: 1972
```

; Routine Size: 32 bytes, Routine Base: \_BAS\$CODE + 03EA

; 950 1973 1

```
952 GLOBAL ROUTINE BASSFIELD_PURGE (      : Purge field variables
953     DECL                                : Scope of the declaration
954 ) : NOVALUE =
955
956
957 **
958 FUNCTIONAL DESCRIPTION:
959     Purge, or undeclare, field variables. This routine is called
960     at the end of a block with declarations that might have been
961     FIELD variables. It scans the symbol table and purges each
962     entry marked as declared in the block.
963
964 FORMAL PARAMETERS:
965     DECL.rl.v      An indication of the scope of the declaration
966                     of the variable. This is a pointer to the major
967                     frame (R11) if the variable is in the scope of
968                     the major procedure, or a pointer to the minor
969                     frame (R10) if the variable is in the scope of
970                     a DEF.
971
972 IMPLICIT INPUTS:
973     SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.
974
975 IMPLICIT OUTPUTS:
976     SYMSQ_ROOT.mq
977
978 ROUTINE VALUE:
979 COMPLETION CODES:
980     NONE
981
982 SIDE EFFECTS:
983     May remove symbols from the symbol table.
984
985
986 --
987 BEGIN
988
989 LOCAL
990     SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
991     SEARCH_DONE;
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
```

```
1009 2031 3 AST_STATUS = $SETAST (ENBFLG = 0);
1010 2032 3
1011 2033 3 IF (.SYMSQ_ROOT [0] EQL 0)
1012 2034 3 THEN
1013 2035 3 BEGIN
1014 2036 3 SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
1015 2037 3 END;
1016 2038 3
1017 2039 3 IF (.AST_STATUS EQL $$$_WASSET) THEN $SETAST (ENBFLG = 1);
1018 2040 3
1019 2041 3 END;
1020 2042 3
1021 2043 3 --+
1022 2044 3 Search the queue, removing any variables declared in this block.
1023 2045 3 --
1024 2046 3 SYM = .SYMSQ_ROOT [0];
1025 2047 3 SEARCH_DONE = 0;
1026 2048 3
1027 2049 3 DO
1028 2050 3 BEGIN
1029 2051 3
1030 2052 3 IF (.SYM EQLA SYMSQ_ROOT)
1031 2053 3 THEN
1032 2054 3 SEARCH_DONE = 1
1033 2055 3 ELSE
1034 2056 3
1035 2057 3 IF (.SYM [SYMSL_DECL] EQL .DECL)
1036 2058 3 THEN
1037 2059 3 BEGIN
1038 2060 3 --+
1039 2061 3 We must delete this symbol from the symbol table.
1040 2062 3 --
1041 2063 3
1042 2064 3 BUILTIN
1043 2065 3 REMQUE;
1044 2066 3
1045 2067 3 LOCAL
1046 2068 3 FREE_VM_STATUS,
1047 2069 3 TEMP;
1048 2070 3 VAR : REF BLOCK [8, BYTE];
1049 2071 3
1050 2072 3 REMQUE (.SYM, TEMP);
1051 2073 3
1052 2074 3 IF (.SYM [SYMSV_INVALID]) THEN BAS$$STOP (BAS$K_ILLFIEVAR);
1053 2075 3
1054 2076 3 VAR = .SYM [SYMSA_VAR];
1055 2077 3 VAR [DSC$W_LENGTH] = 0;
1056 2078 3 VAR [DSC$B_CLASS] = DSC$K_CLASS_D;
1057 2079 3 VAR [DSC$A_POINTER] = 0;
1058 2080 3 FREE_VM_STATUS = LIB$FREE_VM (XREF (SYMSK_LENGTH), TEMP);
1059 2081 3
1060 2082 3 IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP (BAS$K_PROLOSSOR);
1061 2083 3
1062 2084 3 SYM = .SYMSQ_ROOT [0];
1063 2085 3 END
1064 2086 3 ELSE
1065 2087 3 SYM = .SYM [SYMSA_NEXT];
```



: 1066  
: 1067  
: 1068  
: 1069  
: 10702088  
2089  
2090  
2091  
2092M  
M  
M  
M  
MEND  
UNTIL (.SEARCH\_DONE);  
END;

! end of BAS\$FIELD\_PURGE

			007C	00000		.ENTRY	BAS\$FIELD_PURGE, Save R2,R3,R4,R5,R6	1974
56	00000000G	00	9E	00002		MOVAB	BAS\$\$STOP, R6	
55	00000000G	00	9E	00009		MOVAB	SY\$\$SETAST, R5	
54	000000000	EF	9E	00010		MOVAB	SYMSQ_ROOT, R4	
5E		08	C2	00017		SUBL2	#8, SP	
		64	D5	0001A		TSTL	SYMSQ_ROOT	2024
		1D	12	0001C		BNEQ	2\$	
		7E	D4	0001E		CLRL	-(SP)	2031
65		01	FB	00020		CALLS	#1, SY\$\$SETAST	
		64	D5	00023		TSTL	SYMSQ_ROOT	2033
		0A	12	00025		BNEQ	1\$	
51		64	9E	00027		MOVAB	SYMSQ_ROOT, R1	2036
04	A4	51	D0	0002A		MOVL	R1, SYMSQ_ROOT+4	
	64	51	D0	0002E		MOVL	R1, SYMSQ_ROOT	
	09	50	D1	00031	1\$:	CMPL	AST_STATUS, #9	2039
		05	12	00034		BNEQ	2\$	
		01	DD	00036		PUSHL	#1	
65		01	FB	00038		CALLS	#1, SY\$\$SETAST	
52		64	D0	0003B	2\$:	MOVL	SYMSQ_ROOT, SYM	2046
		53	D4	0003E		CLRL	SEARCH_DONE	2047
50		64	9E	00040	3\$:	MOVAB	SYMSQ_ROOT, R0	2052
50		52	D1	00043		CMPL	SYM, R0	
		05	12	00046		BNEQ	4\$	
53		01	D0	00048		MOVL	#1, SEARCH_DONE	2054
		46	11	0004B		BRB	8\$	
04	AC	14	A2	D1 0004D	4\$:	CMPL	20(SYM), DECL	2057
			3C	12 00052		BNEQ	7\$	
04	AE		62	0F 00054		REMQUE	(SYM), TEMP	2072
	07	1C	A2	E9 00058		BLBC	28(SYM), 5\$	2074
	7E	00G	8F	9A 0005C		MOVZBL	#BAS\$K_ILLFIEVAR, -(SP)	
	66		01	FB 00060		CALLS	#1, BAS\$\$STOP	
	50	18	A2	D0 00063	5\$:	MOVL	24(SYM), VAR	2076
			60	B4 00067		CLRW	(VAR)	2077
03	A0		02	90 00069		MOVB	#2, 3(VAR)	2078
		04	A0	D4 0006D		CLRL	4(VAR)	2079
		04	AE	9F 00070		PUSHAB	TEMP	2080
04	AE		20	D0 00073		MOVL	#32, 4(SP)	
		04	AE	9F 00077		PUSHAB	4(SP)	
00000000G	00		02	FB 0007A		CALLS	#2, LIB\$FREE VM	
	07		50	E8 00081		BLBS	FREE VM STATUS, 6\$	2082
	7E	00G	8F	9A 00084		MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	
	66		01	FB 00088		CALLS	#1, BAS\$\$STOP	
	52		64	D0 0008B	6\$:	MOVL	SYMSQ_ROOT, SYM	2084
			03	11 0008E		BRB	8\$	2057
	52		62	D0 00090	7\$:	MOVL	(SYM), SYM	2087
	AA		53	E9 00093	8\$:	BLBC	SEARCH_DONE, 3\$	2090
			04	00096		RET		2092

BASRSTS\_FIELD  
1-023

<sup>3</sup>  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTSFI.B32:1

Page 31  
(9)

; Routine Size: 151 bytes. Routine Base: \_BAS\$CODE + 040A

; 1071 2093 1

```
1073 2094 1 GLOBAL ROUTINE BASSFIELD_OPEN (      ! Account for OPENing a file
1074 2095 1     CHAN                                ! Channel just OPENed
1075 2096 1     ) : NOVALUE =
1076 2097 1
1077 2098 1 ++
1078 2099 1 FUNCTIONAL DESCRIPTION:
1079 2100 1
1080 2101 1     Account for OPENing a file.  If the record length is shorter
1081 2102 1     than before, some variables may have to be un-fielded.
1082 2103 1
1083 2104 1 FORMAL PARAMETERS:
1084 2105 1
1085 2106 1     CHAN.rl.v      The channel number of the file just opened.
1086 2107 1
1087 2108 1 IMPLICIT INPUTS:
1088 2109 1
1089 2110 1     SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.
1090 2111 1
1091 2112 1 IMPLICIT OUTPUTS:
1092 2113 1
1093 2114 1     SYMSQ_ROOT.mq
1094 2115 1
1095 2116 1 ROUTINE VALUE:
1096 2117 1 COMPLETION CODES:
1097 2118 1
1098 2119 1     NONE
1099 2120 1
1100 2121 1 SIDE EFFECTS:
1101 2122 1
1102 2123 1     May remove symbols from the symbol table.
1103 2124 1
1104 2125 1 --
1105 2126 1
1106 2127 1 BEGIN
1107 2128 1
1108 2129 1 BUILTIN
1109 2130 1     FP;
1110 2131 1
1111 2132 1 GLOBAL REGISTER
1112 2133 1     CCB = K_CCB_REG : REF BLOCK [, BYTE];
1113 2134 1
1114 2135 1 LOCAL
1115 2136 1     FMP : REF BLOCK [, BYTE],
1116 2137 1     SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
1117 2138 1     SEARCH_DONE,
1118 2139 1     LUN_NO,
1119 2140 1     RSZ,
1120 2141 1     RBF;
1121 2142 1
1122 2143 1     FMP = .FP;
1123 2144 1
1124 2145 1 ++
1125 2146 1     Compute the logical unit number from the channel number.
1126 2147 1
1127 2148 1     IF (.CHAN LSS 0) THEN BASS$STOP (BASSK_ILLIO_CHA);
1128 2149 1
1129 2150 2     IF (.CHAN EQL 0) THEN LUN_NO = LUB$K_LUN_INPU ELSE LUN_NO = .CHAN;
```



```
1130 2151
1131 2152
1132 2153
1133 2154
1134 2155
1135 2156
1136 2157
1137 2158
1138 2159
1139 2160
1140 2161
1141 2162
1142 2163
1143 2164
1144 2165
1145 2166
1146 2167
1147 2168
1148 2169
1149 2170
1150 2171
1151 2172
1152 2173
1153 2174
1154 2175
1155 2176
1156 2177
1157 2178
1158 2179
1159 2180
1160 2181
1161 2182
1162 2183
1163 2184
1164 2185
1165 2186
1166 2187
1167 2188
1168 2189
1169 2190
1170 2191
1171 2192
1172 2193
1173 2194
1174 2195
1175 2196
1176 2197
1177 2198
1178 2199
1179 2200
1180 2201
1181 2202
1182 2203
1183 2204
1184 2205
1185 2206
1186 2207

+ If the symbol table root has not yet been initialized, initialize it.
-
  IF (.SYMSQ_ROOT [0] EQL 0)
  THEN
    BEGIN
      LOCAL
        AST_STATUS;
      AST_STATUS = $SETAST (ENBFLG = 0);
      IF (.SYMSQ_ROOT [0] EQL 0)
      THEN
        BEGIN
          SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
        END;
      IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
    END;

+ Pick up the buffer size to compare against the variables.
-
  BAS$$CB PUSH (.LUN NO, LUB$K LUN_INPU);
  CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
  RBF = .CCB [LUB$A_RBUF_ADR];
  RSZ = .CCB [LUB$W_RBUF_SIZE];

+ Search the queue, removing any variables which no longer fit in
  the current buffer.
-
  SYM = .SYMSQ_ROOT [0];
  SEARCH_DONE = 0;
  DO
    BEGIN
      IF (.SYM EULA SYMSQ_ROOT)
      THEN
        SEARCH_DONE = 1
      ELSE
        IF (.SYM [SYMSL_CHAN] EQL .CHAN)
        THEN
          BEGIN
            IF (.SYM [SYMSL_OFFSET] + .SYM [SYMSL_LEN] LEQ .RSZ)
            THEN
              BEGIN
+ The variable is still within the buffer, recompute its address,
- since the buffer may have been reallocated.
```

```
1187 2208 5
1188 2209 5
1189 2210 5 LOCAL
1190 2211 5 VAR : REF BLOCK [8, BYTE];
1191 2212 5
1192 2213 5 VAR = .SYM [SYM$A_VAR];
1193 2214 5 VAR [DSC$A_POINTER] = .RBF + .SYM [SYM$L_OFFSET];
1194 2215 5 !+
1195 2216 5 !- Clear the "invalid" bit, since it may have been set by an implied close.
1196 2217 5
1197 2218 5 SYM [SYM$V_INVALID] = 0;
1198 2219 5 SYM = .SYM [SYM$A_NEXT];
1199 2220 5 END
1200 2221 5 ELSE
1201 2222 5 BEGIN
1202 2223 5 !+
1203 2224 5 !- This variable is outside the new buffer, remove it.
1204 2225 5
1205 2226 5 BUILTIN
1206 2227 5 REMQUE;
1207 2228 5
1208 2229 5 LOCAL
1209 2230 5 FREE_VM_STATUS,
1210 2231 5 TEMP;
1211 2232 5 VAR : REF BLOCK [8, BYTE];
1212 2233 5
1213 2234 5 REMQUE (.SYM, TEMP);
1214 2235 5 VAR = .SYM [SYM$A_VAR];
1215 2236 5 VAR [DSC$W_LENGTH] = 0;
1216 2237 5 VAR [DSC$B_CLASS] = DSC$K_CLASS_D;
1217 2238 5 VAR [DSC$A_POINTER] = 0;
1218 2239 5 FREE_VM_STATUS = LIB$FREE_VM (%REF (SYM$K_LENGTH), TEMP);
1219 2240 5
1220 2241 5 IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP (BAS$K_PROLOSSOR);
1221 2242 5
1222 2243 5 SYM = .SYM$Q_ROOT [0];
1223 2244 5 END
1224 2245 5
1225 2246 5 END
1226 2247 5 ELSE
1227 2248 5 SYM = .SYM [SYM$A_NEXT];
1228 2249 5
1229 2250 5 END
1230 2251 5 UNTIL (.SEARCH_DONE);
1231 2252 5
1232 2253 5 !+
1233 2254 5 !- We are through with register CCB.
1234 2255 5
1235 2256 5 BAS$$CB_POP ();
1236 2257 5 END;
```

! end of BAS\$FIELD\_OPEN

09FC 00000

.ENTRY BAS\$FIELD\_OPEN, Save R2,R3,R4,R5,R6,R7,R8,- : 2094  
R11

58	00000000G	00	9E	00002	MOVAB	SYSS\$SETAST, R2	2143
57	00000000G	00	9E	00009	MOVAB	BAS\$\$STOP, R7	2148
56	00000000	EF	9E	00010	MOVAB	SYMSQ_ROOT, R6	
55		08	C2	00017	SUBL2	#8, SP	
53		5D	D0	0001A	MOVL	FP, FMP	
52	04	AC	D0	0001D	MOVL	CHAN, R2	
7E	00G	07	18	00021	BGEQ	1\$	
67		8F	9A	00023	MOVZBL	#BAS\$K_ILLIO CHA, -(SP)	
		01	FB	00027	CALLS	#1, BAS\$\$STOP	
		52	D5	0002A	TSTL	R2	2150
		03	12	0002C	BNEQ	2\$	
52		07	CE	0002E	MNEGL	#7, LUN NO	
		66	D5	00031	TSTL	SYMSQ_ROOT	2156
		1D	12	00033	BNEQ	4\$	
		7E	D4	00035	CLRL	-(SP)	2163
68		01	FB	00037	CALLS	#1, SYSS\$SETAST	2165
		66	D5	0003A	TSTL	SYMSQ_ROOT	
		0A	12	0003C	BNEQ	3\$	2168
51		66	9E	0003E	MOVAB	SYMSQ_ROOT, R1	
04	A6	51	D0	00041	MOVL	R1, SYMSQ_ROOT+4	
66		51	D0	00045	MOVL	R1, SYMSQ_ROOT	
09		50	D1	00048	CMPL	AST_STATUS, #9	2171
		05	12	0004B	BNEQ	4\$	
		01	DD	0004D	PUSHL	#1	
68		01	FB	0004F	CALLS	#1, SYSS\$SETAST	
50		07	CE	00052	MNEGL	#7, R0	2178
	00000000G	00	16	00055	JSB	BAS\$\$CB_PUSH	
FF4C	CB	0C	A3	D0	MOVL	12(FMP), -180(CCB)	2179
	EC	AB	D0	0005B	MOVL	-20(CCB), RBF	2180
54		AB	3C	00065	MOVZWL	-46(CCB), RSZ	2181
55	D2	66	D0	00069	MOVL	SYMSQ_ROOT, SYM	2186
52		53	D4	0006C	CLRL	SEARCH_DONE	2187
		66	9E	0006E	MOVAB	SYMSQ_ROOT, R0	2192
50		52	D1	00071	CMPL	SYM, R0	
		05	12	00074	BNEQ	6\$	
53		01	D0	00076	MOVL	#1, SEARCH_DONE	2194
		56	11	00079	BRB	10\$	
04	AC	08	A2	D1	CMPL	8(SYM), CHAN	2197
		4C	12	00080	BNEQ	9\$	
50	0C	A2	C1	00082	ADDL3	16(SYM), 12(SYM), R0	2201
		50	D1	00088	CMPL	R0, RSZ	
		10	14	0008B	BGTR	7\$	
50		A2	D0	0008D	MOVL	24(SYM), VAR	2212
04	A0	0C	9E	00091	MOVAB	@12(SYM)[RBF], 4(VAR)	2213
1C	A2	01	8A	00097	BICB2	#1, 28(SYM)	2217
		31	11	0009B	BRB	9\$	2218
04	AF	62	0F	0009D	REMQUE	(SYM), TEMP	2234
		A2	D0	000A1	MOVL	24(SYM), VAR	2235
		60	B4	000A5	CLRW	(VAR)	2236
03	A0	02	90	000A7	MOVB	#2, 3(VAR)	2237
		04	A0	D4	CLRL	4(VAR)	2238
		04	AE	9F	PUSHAB	TEMP	2239
04	AE	20	D0	000B1	MOVL	#32, 4(SP)	
		04	AE	9F	PUSHAB	4(SP)	
00000000G	00	02	FB	000B8	CALLS	#2, LIB\$FREE VM	
	07	50	E8	000BF	BLBS	FREE VM STATUS, 8\$	2241
	7E	00G	8F	9A	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	



BASRSTS\_FIELD  
1-023

D 4  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTSF1.B32;1

Page 36  
(10)

67	01	FB	000C6		CALLS	#1, BAS\$\$STOP
52	66	D0	000C9	8\$:	MOVL	SYMSQ_ROOT, SYM
	03	11	000CC		BRB	10\$
52	62	D0	000CE	9\$:	MOVL	(SYM), SYM
9A	53	E9	000D1	10\$:	BLBC	SEARCH DONE, 5\$
	00	16	000D4		JSB	BAS\$\$CB_POP
		04	000DA		RET	

: 2243  
: 2199  
: 2248  
: 2251  
: 2256  
: 2257

; Routine Size: 219 bytes, Routine Base: \_BAS\$CODE + 04A1

; 1237 2258 1

```
1239 2259 1 GLOBAL ROUTINE BASSFIELD_CLOSE (      ! Account for CLOSEing a file
1240 2260 1     CHAN                                ! Channel about to be CLOSEed
1241 2261 1     ) : NOVALUE =
1242 2262 1
1243 2263 1 ++
1244 2264 1 FUNCTIONAL DESCRIPTION:
1245 2265 1
1246 2266 1     Account for CLOSEing a file. Unfield all of the variables
1247 2267 1     on this channel.
1248 2268 1
1249 2269 1 FORMAL PARAMETERS:
1250 2270 1
1251 2271 1     CHAN.rl.v      The channel number of the file about to be
1252 2272 1                   CLOSEed.
1253 2273 1
1254 2274 1 IMPLICIT INPUTS:
1255 2275 1
1256 2276 1     SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.
1257 2277 1
1258 2278 1 IMPLICIT OUTPUTS:
1259 2279 1
1260 2280 1     SYMSQ_ROOT.mq
1261 2281 1     LUBSV_FIELD_USE for this channel, set to 0
1262 2282 1
1263 2283 1 ROUTINE VALUE:
1264 2284 1 COMPLETION CODES:
1265 2285 1
1266 2286 1     NONE
1267 2287 1
1268 2288 1 SIDE EFFECTS:
1269 2289 1
1270 2290 1     May remove symbols from the symbol table.
1271 2291 1
1272 2292 1 --
1273 2293 1
1274 2294 2 BEGIN
1275 2295 2
1276 2296 2 GLOBAL REGISTER
1277 2297 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
1278 2298 2
1279 2299 2 LOCAL
1280 2300 2     SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BASSFIELD_SYM),
1281 2301 2     LUN_NO,
1282 2302 2     SEARCH_DONE;
1283 2303 2
1284 2304 2 ++
1285 2305 2     If the symbol table root has not yet been initialized, initialize it.
1286 2306 2 --
1287 2307 2
1288 2308 2 IF (.SYMSQ_ROOT [0] EQL 0)
1289 2309 2 THEN
1290 2310 2     BEGIN
1291 2311 2
1292 2312 2     LOCAL
1293 2313 2         AST_STATUS;
1294 2314 2
1295 2315 2     AST_STATUS = $SETAST (ENBFLG = 0);
```

```
.. 1296      2316      3
.. 1297      2317
.. 1298      2318
.. 1299      2319
.. 1300      2320      IF (.SYMSQ_ROOT [0] EQL 0)
.. 1301      2321      THEN
.. 1302      2322      BEGIN
.. 1303      2323      SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
.. 1304      2324      END;
.. 1305      2325      IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
.. 1306      2326      END;
.. 1307      2327      +
.. 1308      2328      Search the queue, removing any variables for this channel.
.. 1309      2329      -
.. 1310      2330      SYM = .SYMSQ_ROOT [0];
.. 1311      2331      SEARCH_DONE = 0;
.. 1312      2332
.. 1313      2333      DO
.. 1314      2334      BEGIN
.. 1315      2335
.. 1316      2336      IF (.SYM EQLA SYMSQ_ROOT)
.. 1317      2337      THEN
.. 1318      2338      SEARCH_DONE = 1
.. 1319      2339      ELSE
.. 1320      2340
.. 1321      2341      IF (.SYM [SYMSL_CHAN] EQL .CHAN)
.. 1322      2342      THEN
.. 1323      2343      BEGIN
.. 1324      2344      +
.. 1325      2345      We must delete this symbol from the symbol table.
.. 1326      2346      -
.. 1327      2347
.. 1328      2348      BUILTIN
.. 1329      2349      REMQUE;
.. 1330      2350
.. 1331      2351      LOCAL
.. 1332      2352      FREE_VM_STATUS,
.. 1333      2353      TEMP;
.. 1334      2354      VAR : REF BLOCK [8, BYTE];
.. 1335      2355
.. 1336      2356      REMQUE (.SYM, TEMP);
.. 1337      2357
.. 1338      2358      IF (.SYM [SYMSV_INVALID]) THEN BAS$$STOP (BAS$K_ILLFIEVAR);
.. 1339      2359
.. 1340      2360      VAR = .SYM [SYMSA_VAR];
.. 1341      2361      VAR [DSC$W_LENGTH] = 0;
.. 1342      2362      VAR [DSC$B_CLASS] = DSC$K_CLASS_D;
.. 1343      2363      VAR [DSC$A_POINTER] = 0;
.. 1344      2364      FREE_VM_STATUS = LIB$FREE_VM (%REF (SYMSK_LENGTH), TEMP);
.. 1345      2365
.. 1346      2366      IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP (BAS$K_PROLOSSOR);
.. 1347      2367
.. 1348      2368      SYM = .SYMSQ_ROOT [0];
.. 1349      2369      END
.. 1350      2370      ELSE
.. 1351      2371      SYM = .SYM [SYMSA_NEXT];
.. 1352      2372
```



```
1353 2373 3      END
1354 2374 3      UNTIL (.SEARCH_DONE);
1355 2375 3
1356 2376 3      +
1357 2377 3      - load register CCB.
1358 2378 3
1359 2379 3      LUN_NO = (IF (.CHAN EQL 0) THEN LUBSK_LUN_INPU ELSE .CHAN);
1360 2380 3      BAS$CB_PUSH (.LUN_NO, LUBSK_ILUN_MINT);
1361 2381 3
1362 2382 3      +
1363 2383 3      - indicate there is not FIELDing on this channel anymore.
1364 2384 3
1365 2385 3      CCB [LUB$V_FIELD_USE] = 0;
1366 2386 3
1367 2387 3      +
1368 2388 3      - done with register CCB.
1369 2389 3
1370 2390 3      BAS$CB_POP ();
1371 2391 3
1372 2392 3      END;

! end of BAS$FIELD_CLOSE
```

			083C 00000	.ENTRY	BAS\$FIELD_CLOSE, Save R2,R3,R4,R5,R11	2259
55	00000000G	00	9E 00002	MOVAB	BAS\$STOP, R5	
54	00000000G	00	9E 00009	MOVAB	SY\$SETAST, R4	
53	00000000'	EF	9E 00010	MOVAB	SYM\$Q_ROOT, R3	
5E		08	C2 00017	SUBL2	#8, SP	
		63	D5 0001A	TSTL	SYM\$Q_ROOT	2308
		1D	12 0001C	BNEQ	2\$	
		7E	D4 0001E	CLRL	-(SP)	2315
64		01	FB 00020	CALLS	#1, SY\$SETAST	
		63	D5 00023	TSTL	SYM\$Q_ROOT	2317
		0A	12 00025	BNEQ	1\$	
04	51	63	9E 00027	MOVAB	SYM\$Q_ROOT, R1	2320
	A3	51	D0 0002A	MOVL	R1, SYM\$Q_ROOT+4	
	63	51	D0 0002E	MOVL	R1, SYM\$Q_ROOT	
	09	50	D1 00031	CMPL	AST_STATUS, #9	2323
		05	12 00034	BNEQ	2\$	
		01	DD 00036	PUSHL	#1	
64		01	FB 00038	CALLS	#1, SY\$SETAST	
52		63	D0 0003B	MOVL	SYM\$Q_ROOT, SYM	2330
		5B	D4 0003E	CLRL	SEARCH_DONE	2331
50		63	9E 00040	MOVAB	SYM\$Q_ROOT, R0	2336
50		52	D1 00043	CMPL	SYM, R0	
		05	12 00046	BNEQ	4\$	
5B		01	D0 00048	MOVL	#1, SEARCH_DONE	2338
		46	11 0004B	BRB	8\$	
04	AC	08	A2 D1 0004D	CMPL	8(SYM), CHAN	2341
		3C	12 00052	BNEQ	7\$	
04	AE	62	0F 00054	REMQUE	(SYM), TEMP	2356
	07	1C	A2 E9 00058	BLBC	28(SYM), 5\$	2358
	7E	00G	8F 9A 0005C	MOVZBL	#BAS\$K_ILLFIEVAR, -(SP)	
	65	01	FB 00060	CALLS	#1, BAS\$STOP	
	50	1B	A2 D0 00063	MOVL	24(SYM), VAR	2360

BASRSTS\_FIELD  
1-023

H 4  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTSFI.B32;1

Page 40  
(11)

03	A0		60	B4	00067	CLRW	(VAR)	..	2361	
			02	90	00069	MOVB	#2, 3(VAR)	...	2362	
		04	A0	D4	0006D	CLRL	4(VAR)	...	2363	
		04	AE	9F	00070	PUSHAB	TEMP	...	2364	
04	AE		20	D0	00073	MOVL	#32, 4(SP)			
		04	AE	9F	00077	PUSHAB	4(SP)			
00000000G	00		02	FB	0007A	CALLS	#2, LIB\$FREE VM			
	07		50	EB	00081	BLBS	FREE VM STATUS, 68	...	2366	
	7E	00G	8F	9A	00084	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)			
	65		01	FB	00088	CALLS	#1, BAS\$\$STOP			
	52		63	D0	0008B	68:	MOVL	SYMSQ_ROOT, SYM	...	2368
			03	11	0008E	BRB	88	...	2341	
	52		62	D0	00090	78:	MOVL	(SYM), SYM	...	2371
	AA		5B	E9	00093	88:	BLBC	SEARCH_DONE, 38	...	2374
		04	AC	D5	00096		TSTL	CHAN	...	2379
			05	12	00099		BNEQ	98		
	52		07	CE	0009B		MNEGL	#7, LUN_NO		
			04	11	0009E		BRB	108		
	52	04	AC	D0	000A0	98:	MOVL	CHAN, LUN_NO		
	50		08	CE	000A4	108:	MNEGL	#8, R0	...	2380
		00000000G	00	16	000A7		JSB	BAS\$\$CB_PUSH		
A1	AB	40	8F	8A	000AD		BICB2	#64, -95(CCB)	...	2385
		00000000G	00	16	000B2		JSB	BAS\$\$CB_POP	...	2390
			04	000B8			RET	...	2392	

; Routine Size: 185 bytes. Routine Base: \_BAS\$CODE + 057C

; 1373 2393 1

```
1375 2394 1 ROUTINE BASSFIELD KILL
1376 2395 1 : CALL_CCB NOVXUE =
1377 2396 1
1378 2397 1
1379 2398 1
1380 2399 1
1381 2400 1
1382 2401 1
1383 2402 1
1384 2403 1
1385 2404 1
1386 2405 1
1387 2406 1
1388 2407 1
1389 2408 1
1390 2409 1
1391 2410 1
1392 2411 1
1393 2412 1
1394 2413 1
1395 2414 1
1396 2415 1
1397 2416 1
1398 2417 1
1399 2418 1
1400 2419 1
1401 2420 1
1402 2421 1
1403 2422 1
1404 2423 1
1405 2424 1
1406 2425 1
1407 2426 1
1408 2427 1
1409 2428 1
1410 2429 1
1411 2430 1
1412 2431 1
1413 2432 1
1414 2433 1
1415 2434 1
1416 2435 1
1417 2436 1
1418 2437 1
1419 2438 1
1420 2439 1
1421 2440 1
1422 2441 1
1423 2442 2
1424 2443 2
1425 2444 2
1426 2445 2
1427 2446 2
1428 2447 2
1429 2448 2
1430 2449 2
1431 2450 2
```

ROUTINE BASSFIELD KILL  
: CALL\_CCB NOVXUE =

++  
FUNCTIONAL DESCRIPTION:

This routine is called while a file is being CLOSED, for any reason. If the CLOSE was explicit and in the module containing the FIELD statement(s), BASSFIELD\_CLOSE will already have removed all of the field variables for this channel from the symbol table, so this routine will find none. If the CLOSE is implicit or outside the module with the FIELD statement(s), BASSFIELD\_CLOSE will not have been called and this routine will mark some variables invalid. An explicit CLOSE from another module is considered a programming error, so it is proper to give an error as soon as any of these variables are referenced. We cannot signal an error from here because this may be the CLOSE from the exit handler (in which case the variables will not be referenced again, so marking them invalid is OK) or the implicit CLOSE from OPEN, in which case (if the OPEN is from a module with FIELD) BASSFIELD\_OPEN will re-validate the variables still in the buffer.

FORMAL PARAMETERS:

NONE

IMPLICIT INPUTS:

SYMSG\_ROOT.mq The queue of FIELD variables : the symbol table.  
LUBSW\_LUN The logical unit number of the file being closed

IMPLICIT OUTPUTS:

SYMSG\_ROOT.mq

ROUTINE VALUE:

COMPLETION CODES:

NONE

SIDE EFFECTS:

May mark symbols invalid, but is most likely to have no net effect.

--

BEGIN

EXTERNAL REGISTER  
CCB : REF BLOCK [, BYTE];

LOCAL  
SYM : REF BLOCK [SYMSK\_LENGTH, BYTE] FIELD (BASSFIELD\_SYM),  
SEARCH\_DONE,  
CHAN;

```
1432 2451
1433 2452
1434 2453
1435 2454
1436 2455
1437 2456
1438 2457
1439 2458
1440 2459
1441 2460
1442 2461
1443 2462
1444 2463
1445 2464
1446 2465
1447 2466
1448 2467
1449 2468
1450 2469
1451 2470
1452 2471
1453 2472
1454 2473
1455 2474
1456 2475
1457 2476
1458 2477
1459 2478
1460 2479
1461 2480
1462 2481
1463 2482
1464 2483
1465 2484
1466 2485
1467 2486
1468 2487
1469 2488
1470 2489
1471 2490
1472 2491
1473 2492
1474 2493
1475 2494
1476 2495
1477 2496
1478 2497
1479 2498
1480 2499
1481 2500
1482 2501
1483 2502
1484 2503
1485 2504
1486 2505
1487 2506
1488 2507

+ If the symbol table root has not yet been initialized, initialize it.
-
IF (.SYMSQ_ROOT [0] EQL 0)
THEN
  BEGIN
    LOCAL
      AST_STATUS;
    AST_STATUS = $SETAST (ENBFLG = 0);
    IF (.SYMSQ_ROOT [0] EQL 0)
    THEN
      BEGIN
        SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
      END;
    IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
  END;

+ Compute the channel number from the logical unit number.
-
CHAN = (IF (.CCB [LUBSW_LUN] EQL LUBSK_LUN_INPU) THEN 0 ELSE .CCB [LUBSW_LUN]);

+ Search the queue, invalidating any variables for this channel.
-
SYM = .SYMSQ_ROOT [0];
SEARCH_DONE = 0;
DO
  BEGIN
    IF (.SYM EQLA SYMSQ_ROOT)
    THEN
      SEARCH_DONE = 1
    ELSE
      IF (.SYM [SYMSL_CHAN] EQL .CHAN)
      THEN
        BEGIN
+ We must mark this symbol as invalid.
-
          LOCAL
            VAR : REF BLOCK [8, BYTE];
          VAR = .SYM [SYMSA_VAR];
          VAR [DSCSA_POINTER] = 0;
          SYM [SYMSV_INVALID] = 1;
        END;
```



BASRSTS\_FIELD  
1-023

K 4  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 BlISS-32 V4.0-742  
[BASRTL.SRC]BASRSTSFI.B32:1

Page 43  
(12)

```
: 1489      2508 3      SYM = .SYM [SYMSA_NEXT];  
: 1490      2509 3      END  
: 1491      2510 3      UNTIL (.SEARCH_DONE);  
: 1492      2511 3  
: 1493      2512 1      END;
```

! end of BAS\$FIELD\_KILL

```
003C 00000 BAS$FIELD_KILL:  
55 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5 2394  
54 00000000' EF 9E 00009 MOVAB SYS$SETAST, R5  
64 D5 00010 MOVAB SYMSQ_ROOT, R4 2456  
1D 12 00012 TSTL SYMSQ_ROOT  
7E D4 00014 BNEQ 2$ 2463  
65 01 FB 00016 CLRL -(SP)  
64 D5 00019 CALLS #1, SYS$SETAST  
0A 12 0001B TSTL SYMSQ_ROOT 2465  
51 64 9E 0001D BNEQ 1$  
04 A4 51 D0 00020 MOVAB SYMSQ_ROOT, R1 2468  
64 51 D0 00024 MOVL R1, SYMSQ_ROOT+4  
09 50 D1 00027 1$: MOVL R1, SYMSQ_ROOT  
05 12 0002A CMPL ASF_STATUS, #9 2471  
01 DD 0002C BNEQ 2$  
FFF9 65 01 FB 0002E CALLS #1, SYS$SETAST  
8F C6 AB B1 00031 2$: CMPW -58(CCB), #-7 2478  
04 12 00037 BNEQ 3$  
52 D4 00039 CLRL CHAN  
04 11 0003B BRB 4$  
52 C6 AB 32 0003D 3$: CVTWL -58(CCB), CHAN  
51 64 D0 00041 4$: MOVL SYMSQ_ROOT, SYM 2482  
53 D4 00044 CLRL SEARCH_DONE 2483  
50 64 9E 00046 5$: MOVAB SYMSQ_ROOT, R0 2488  
50 51 D1 00049 CMPL SYM, R0  
05 12 0004C BNEQ 6$  
53 01 D0 0004E MOVL #1, SEARCH_DONE 2490  
11 11 00051 BRB 7$  
52 08 A1 D1 00053 6$: CMPL 8(SYM), CHAN 2493  
0B 12 00057 BNEQ 7$  
50 18 A1 D0 00059 MOVL 24(SYM), VAR 2503  
04 A0 D4 0005D CLRL 4(VAR) 2504  
1C A1 01 88 00060 BISB2 #1, 28(SYM) 2505  
51 61 D0 00064 7$: MOVL (SYM), SYM 2508  
DC 53 E9 00067 BLBC SEARCH_DONE, 5$ 2510  
04 0006A RET 2512
```

; Routine Size: 107 bytes, Routine Base: \_BAS\$CODE + 0635

; 1494 2513 1

```
1496 2514 GLOBAL ROUTINE BAS$FIELD_INIT : NOVALUE =      ! Initialize for RUN
1497 2515
1498 2516
1499 2517
1500 2518
1501 2519
1502 2520
1503 2521
1504 2522
1505 2523
1506 2524
1507 2525
1508 2526
1509 2527
1510 2528
1511 2529
1512 2530
1513 2531
1514 2532
1515 2533
1516 2534
1517 2535
1518 2536
1519 2537
1520 2538
1521 2539
1522 2540
1523 2541
1524 2542
1525 2543
1526 2544
1527 2545
1528 2546
1529 2547
1530 2548
1531 2549
1532 2550
1533 2551
1534 2552
1535 2553
1536 2554
1537 2555
1538 2556
1539 2557
1540 2558
1541 2559
1542 2560
1543 2561
1544 2562
1545 2563
1546 2564
1547 2565
1548 2566
1549 2567
1550 2568
1551 2569
1552 2570

GLOBAL ROUTINE BAS$FIELD_INIT : NOVALUE =      ! Initialize for RUN

++
FUNCTIONAL DESCRIPTION:
    Initialize the FIELD symbol table for the RUN command.  All symbols are removed
    from the table, even those marked invalid.

FORMAL PARAMETERS:
    NONE

IMPLICIT INPUTS:
    SYMSQ_ROOT.mq  The queue of FIELD variables : the symbol table.

IMPLICIT OUTPUTS:
    SYMSQ_ROOT.mq

ROUTINE VALUE:
COMPLETION CODES:
    NONE

SIDE EFFECTS:
    Makes the symbol table empty.

--
BEGIN
LOCAL
SYM : REF BLOCK [SYMSK_LENGTH, BYTE] FIELD (BAS$FIELD_SYM),
SEARCH_DONE;

++
If the symbol table root has not yet been initialized, initialize it.
--
IF (.SYMSQ_ROOT [0] EQL 0)
THEN
BEGIN
LOCAL
AST_STATUS;
AST_STATUS = $SETAST (ENBFLG = 0);
IF (.SYMSQ_ROOT [0] EQL 0)
THEN
BEGIN
SYMSQ_ROOT [0] = SYMSQ_ROOT [1] = SYMSQ_ROOT [0];
END;
IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
```

```
1553 2571 3
1554 2572 3
1555 2573 3
1556 2574 3
1557 2575 3
1558 2576 3
1559 2577 3
1560 2578 3
1561 2579 3
1562 2580 3
1563 2581 3
1564 2582 3
1565 2583 4
1566 2584 3
1567 2585 3
1568 2586 3
1569 2587 4
1570 2588 4
1571 2589 4
1572 2590 4
1573 2591 4
1574 2592 4
1575 2593 4
1576 2594 4
1577 2595 4
1578 2596 4
1579 2597 4
1580 2598 4
1581 2599 4
1582 2600 4
1583 2601 4
1584 2602 4
1585 2603 4
1586 2604 4
1587 2605 4
1588 2606 4
1589 2607 4
1590 2608 4
1591 2609 3
1592 2610 2
1593 2611 2
1594 2612 1

END;

+ Search the queue, deleting any symbols in it.
-
SYM = .SYM$Q_ROOT [0];
SEARCH_DONE = 0;

DO
  BEGIN
    IF (.SYM EQLA SYM$Q_ROOT)
    THEN
      SEARCH_DONE = 1
    ELSE
      BEGIN
+ We must delete this symbol from the symbol table.
-
        BUILTIN
          REMQUE;

        LOCAL
          FREE_VM_STATUS,
          TEMP;
          VAR : REF BLOCK [8, BYTE];

        REMQUE (.SYM, TEMP);
        VAR = .SYM [SYM$A_VAR];
        FREE_VM_STATUS = [IB$FREE_VM (%REF (SYM$K_LENGTH), TEMP);

        IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP (BAS$K_PROLOSSOR);

        SYM = .SYM$Q_ROOT [0];
      END
    END
  UNTIL (.SEARCH_DONE);

END;
```

! end of BAS\$\$FIELD\_INIT

```
55 00000000G 003C 00000
54 00000000' 00 9E 00002
5E 08 C2 00010
64 D5 00013
1D 12 00015
7E D4 00017
65 01 FB 00019
64 D5 0001C
0A 12 0001E
```

```
.ENTRY BAS$$FIELD_INIT, Save R2,R3,R4,R5
MOVAB SY$$SETAST, R5
MOVAB SYM$Q_ROOT, R4
SUBL2 #8, SP
TSTL SYM$Q_ROOT
BNEQ 2$
CLRL -(SP)
CALLS #1, SY$$SETAST
TSTL SYM$Q_ROOT
BNEQ 1$
```

```
: 2514
:
:
: 2555
:
: 2562
:
: 2564
:
```



BASSRSTS\_FIELD  
1-023

N 4  
16-Sep-1984 01:07:30  
14-Sep-1984 11:56:38

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRSTSFI.B32;1

Page 46  
(13)

04	51	64	9E	00020	MOVAB	SYMSQ ROOT, R1	2567
	A4	51	D0	00023	MOVL	R1, SYMSQ_ROOT+4	
	64	51	D0	00027	MOVL	R1, SYMSQ_ROOT	
	09	50	D1	0002A	1\$: CMPL	AST_STATUS, #9	2570
		05	12	0002D	BNEQ	2\$	
	65	01	DD	0002F	PUSHL	#1	
	52	01	FB	00031	CALLS	#1, SYSSSETAST	
		64	D0	00034	2\$: MOVL	SYMSQ_ROOT, SYM	2577
	50	53	D4	00037	CLRL	SEARCH_DONE	2578
	50	64	9E	00039	3\$: MOVAB	SYMSQ_ROOT, R0	2583
		52	D1	0003C	CMPL	SYM, R0	
	53	05	12	0003F	BNEQ	4\$	
		01	D0	00041	MOVL	#1, SEARCH_DONE	2585
	AE	2A	11	00044	BRB	6\$	
04	50	62	0F	00046	4\$: REMQUE	(SYM), TEMP	2600
		A2	D0	0004A	MOVL	24(SYM), VAR	2601
	AE	AE	9F	0004E	PUSHAB	TEMP	2602
04		20	D0	00051	MOVL	#32, 4(SP)	
	00	AE	9F	00055	PUSHAB	4(SP)	
00000000G	0B	02	FB	00058	CALLS	#2, LIB\$FREE VM	
	7E	50	E8	0005F	BLBS	FREE VM STATUS, 5\$	2604
00000000G	00	8F	9A	00062	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	
	52	01	FB	00066	CALLS	#1, BAS\$\$STOP	
	C6	64	D0	0006D	5\$: MOVL	SYMSQ_ROOT, SYM	2606
		53	E9	00070	6\$: BLBC	SEARCH_DONE, 3\$	2610
		04	00073	RET			2612

; Routine Size: 116 bytes. Routine Base: \_BAS\$CODE + 06A0

: 1595 2613 1  
: 1596 2614 1 END  
: 1597 2615 1  
: 1598 2616 0 ELUDOM

! end of module BASSRSTS\_FIELD

#### PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$DATA	8	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_BAS\$CODE	1812	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

#### Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_S255\$DUA28:[SYSLIB]STARLET.L32;1	9776	12	0	581	00:01.2



COMMAND QUALIFIERS

```
:  
:      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASRSTSFI/OBJ=OBJ$:BASRSTSFI MSRC$:BASRSTSFI/UPDATE=(ENH$:BASRSTSFI  
:  
: Size:      1812 code + 8 data bytes  
: Run Time:   00:40.2  
: Elapsed Time: 01:22.3  
: Lines/CPU Min: 3900  
: Lexemes/CPU-Min: 25164  
: Memory Used: 227 pages  
: Compilation Complete
```



0031

AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY